SwiftUI / Search

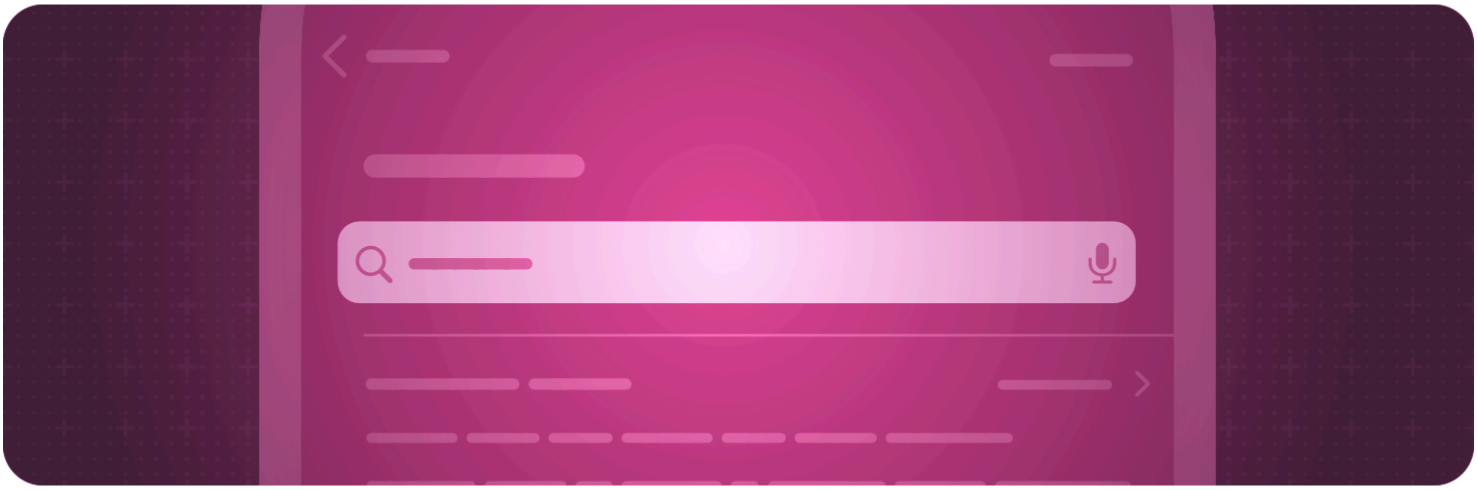
API Collection

Search

Enable people to search for text or other content within your app.

Overview

To present a search field in your app, create and manage storage for search text and optionally for discrete search terms known as *tokens*. Then bind the storage to the search field by applying the searchable view modifier to a view in your app.



As people interact with the field, they implicitly modify the underlying storage and, thereby, the search parameters. Your app correspondingly updates other parts of its interface. To enhance the search interaction, you can also:

- Offer suggestions during search, for both text and tokens.
- Implement search scopes that help people to narrow the search space.
- Detect when people activate the search field, and programmatically dismiss the search field using environment values.

For design guidance, see Searching in the Human Interface Guidelines.

Topics

Searching your app’s data model

Adding a search interface to your app

Present an interface that people can use to search for content in your app.

Performing a search operation

Update search results based on search text and optional tokens that you store.

```
func searchable(text:placement:prompt:)
```

Marks this view as searchable, which configures the display of a search field.

```
func searchable(text:tokens:placement:prompt:token:)
```

Marks this view as searchable with text and tokens.

```
func searchable(text:editableTokens:placement:prompt:token:)
```

Marks this view as searchable, which configures the display of a search field.

```
struct SearchFieldPlacement
```

The placement of a search field in a view hierarchy.

Making search suggestions

Suggesting search terms

Provide suggestions to people searching for content in your app.

```
func searchSuggestions<S>(() -> S) -> some View
```

Configures the search suggestions for this view.

```
func searchSuggestions(Visibility, for: SearchSuggestionsPlacement.Set)
-> some View
```

Configures how to display search suggestions within this view.

```
func searchCompletion(_:)
```

Associates a fully formed string with the value of this view when used as a search suggestion.

```
func searchable(text:tokens:suggestedTokens:placement:prompt:token:)
```

Marks this view as searchable with text, tokens, and suggestions.

```
struct SearchSuggestionsPlacement
```

The ways that SwiftUI displays search suggestions.

Limiting search scope

Scoping a search operation

Divide the search space into a few broad categories.

```
func searchScopes<V, S>(Binding<V>, scopes: () -> S) -> some View
```

Configures the search scopes for this view.

```
func searchScopes<V, S>(Binding<V>, activation: SearchScopeActivation,
() -> S) -> some View
```

Configures the search scopes for this view with the specified activation strategy.

```
struct SearchScopeActivation
```

The ways that searchable modifiers can show or hide search scopes.

Detecting, activating, and dismissing search

Managing search interface activation

Programmatically detect and dismiss a search field.

```
var isSearching: Bool
```

A Boolean value that indicates when the user is searching.

```
var dismissSearch: DismissSearchAction
```

An action that ends the current search interaction.

```
struct DismissSearchAction
```

An action that can end a search interaction.

```
func searchable(text:isPresented:placement:prompt:)
```

Marks this view as searchable with programmatic presentation of the search field.

```
func searchable(text:tokens:isPresented:placement:prompt:token:)
```

Marks this view as searchable with text and tokens, as well as programmatic presentation.

```
func searchable(text:editableTokens:isPresented:placement:prompt:token
:)
```

Marks this view as searchable, which configures the display of a search field.

`func searchable(text:tokens:suggestedTokens:isPresented:placement:prompt:token:)`

Marks this view as searchable with text, tokens, and suggestions, as well as programmatic presentation.

Displaying toolbar content during search

`func searchPresentationToolbarBehavior(SearchPresentationToolbarBehavior) -> some View`

Configures the search toolbar presentation behavior for any searchable modifiers within this view.

`struct SearchPresentationToolbarBehavior`

A type that defines how the toolbar behaves when presenting search.

Searching for text in a view

`func findNavigator(isPresented: Binding<Bool>) -> some View`

Programmatically presents the find and replace interface for text editor views.

`func findDisabled(Bool) -> some View`

Prevents find and replace operations in a text editor.

`func replaceDisabled(Bool) -> some View`

Prevents replace operations in a text editor.

`struct FindContext`

The status of the find navigator for views which support text editing.

See Also

App structure

App organization

Define the entry point and top-level structure of your app.

Scenes

Declare the user interface groupings that make up the parts of your app.

Windows

Display user interface content in a window or a collection of windows.

Immersive spaces

Display unbounded content in a person’s surroundings.

Documents

Enable people to open and manage documents.

Navigation

Enable people to move between different parts of your app’s view hierarchy within a scene.

Modal presentations

Present content in a separate view that offers focused interaction.

Toolbars

Provide immediate access to frequently used commands and controls.

App extensions

Extend your app’s basic functionality to other parts of the system, like by adding a Widget.