

Framework

SwiftUI

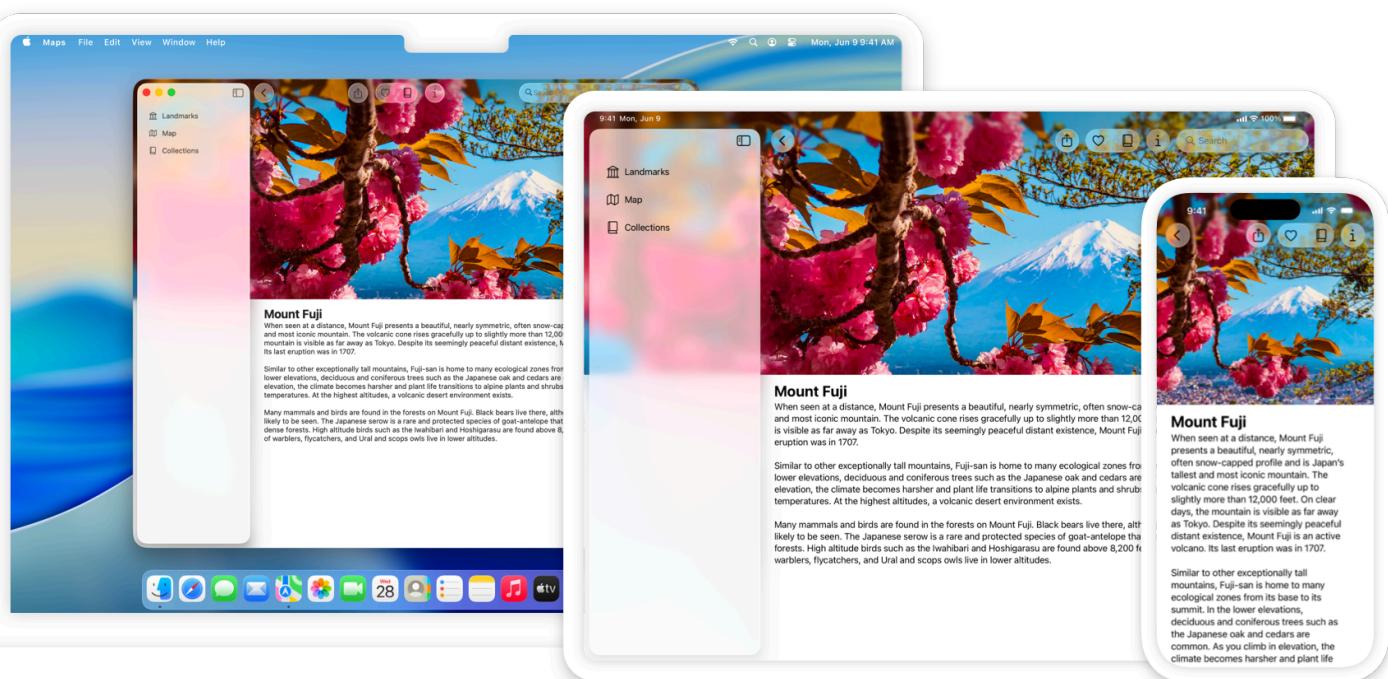
Declare the user interface and behavior for your app on every platform.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

Overview

SwiftUI provides views, controls, and layout structures for declaring your app's user interface. The framework provides event handlers for delivering taps, gestures, and other types of input to your app, and tools to manage the flow of data from your app's models down to the views and controls that users see and interact with.

Define your app structure using the [App protocol](#), and populate it with scenes that contain the views that make up your app's user interface. Create your own custom views that conform to the [View protocol](#), and compose them with SwiftUI views for displaying text, images, and custom shapes using stacks, lists, and more. Apply powerful modifiers to built-in views and your own views to customize their rendering and interactivity. Share code between apps on multiple platforms with views and controls that adapt to their context and presentation.



You can integrate SwiftUI views with objects from the [UIKit](#), [AppKit](#), and [WatchKit](#) frameworks to take further advantage of platform-specific functionality. You can also customize accessibility support in SwiftUI, and localize your app's interface for different languages, countries, or cultural regions.

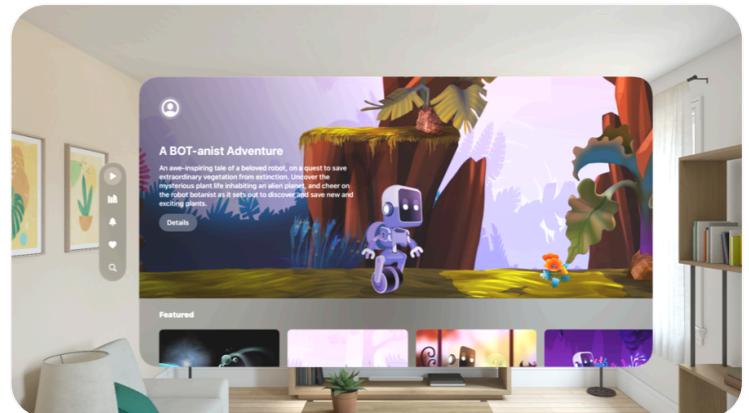
Featured samples



Landmarks: Building an app with Liquid Glass

Enhance your app experience with system-provided and custom Liquid Glass.

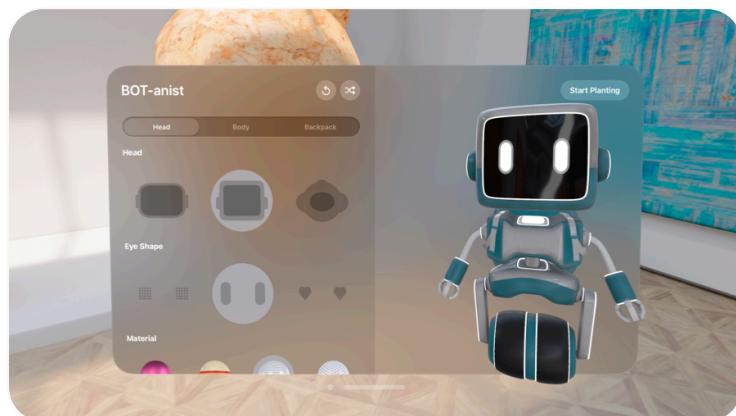
[View sample code >](#)



Destination Video

Leverage SwiftUI to build an immersive media experience in a multiplatform app.

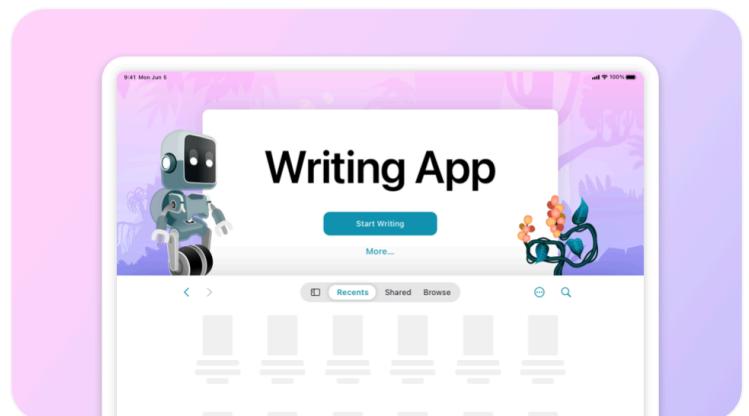
[View sample code >](#)



BOT-anist

Build a multiplatform app that uses windows, volumes, and animations to create a robot botanist's greenhouse.

[View sample code >](#)



Building a document-based app with SwiftUI

Create, save, and open documents in a multiplatform app.

[View sample code >](#)

Topics

Essentials

Adopting Liquid Glass

Find out how to bring the new material to your app.

Learning SwiftUI

Discover tips and techniques for building multiplatform apps with this set of conceptual articles and sample code.

Exploring SwiftUI Sample Apps

Explore these SwiftUI samples using Swift Playgrounds on iPad or in Xcode to learn about defining user interfaces, responding to user interactions, and managing data flow.

SwiftUI updates

Learn about important changes to SwiftUI.

Landmarks: Building an app with Liquid Glass

Enhance your app experience with system-provided and custom Liquid Glass.

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.

Search

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

App extensions

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

Data and storage

Model data

Manage the data that your app uses to drive its interface.

Environment values

Share data throughout a view hierarchy using the environment.

Preferences

Indicate configuration preferences from views to their container views.

Persistent storage

Store data for use across sessions of your app.

Views

View fundamentals

Define the visual elements of your app using a hierarchy of views.

View configuration

Adjust the characteristics of views in a hierarchy.

View styles

Apply built-in and custom appearances and behaviors to different types of views.

Animations

Create smooth visual updates in response to state changes.

Text input and output

Display formatted text and get text input from the user.

- ☰ Images
 - Add images and symbols to your app's user interface.
- ☰ Controls and indicators
 - Display values and get user selections.
- ☰ Menus and commands
 - Provide space-efficient, context-dependent access to commands and controls.
- ☰ Shapes
 - Trace and fill built-in and custom shapes with a color, gradient, or other pattern.
- ☰ Drawing and graphics
 - Enhance your views with graphical effects and customized drawings.

View layout

- ☰ Layout fundamentals
 - Arrange views inside built-in layout containers like stacks and grids.
- ☰ Layout adjustments
 - Make fine adjustments to alignment, spacing, padding, and other layout parameters.
- ☰ Custom layout
 - Place views in custom arrangements and create animated transitions between layout types.
- ☰ Lists
 - Display a structured, scrollable column of information.
- ☰ Tables
 - Display selectable, sortable data arranged in rows and columns.
- ☰ View groupings
 - Present views in different kinds of purpose-driven containers, like forms or control groups.
- ☰ Scroll views
 - Enable people to scroll to content that doesn't fit in the current display.

Event handling

- ☰ Gestures

Define interactions from taps, clicks, and swipes to fine-grained gestures.

☰ Input events

Respond to input from a hardware device, like a keyboard or a Touch Bar.

☰ Clipboard

Enable people to move or duplicate items by issuing Copy and Paste commands.

☰ Drag and drop

Enable people to move or duplicate items by dragging them from one location to another.

☰ Focus

Identify and control which visible object responds to user interaction.

☰ System events

React to system events, like opening a URL.

Accessibility

☰ Accessibility fundamentals

Make your SwiftUI apps accessible to everyone, including people with disabilities.

☰ Accessible appearance

Enhance the legibility of content in your app's interface.

☰ Accessible controls

Improve access to actions that your app can undertake.

☰ Accessible descriptions

Describe interface elements to help people understand what they represent.

☰ Accessible navigation

Enable users to navigate to specific user interface elements using rotors.

Framework integration

☰ AppKit integration

Add AppKit views to your SwiftUI app, or use SwiftUI views in your AppKit app.

☰ UIKit integration

Add UIKit views to your SwiftUI app, or use SwiftUI views in your UIKit app.

WatchKit integration

Add WatchKit views to your SwiftUI app, or use SwiftUI views in your WatchKit app.

Technology-specific views

Use SwiftUI views that other Apple frameworks provide.

Tool support

Previews in Xcode

Generate dynamic, interactive previews of your custom views.

Xcode library customization

Expose custom views and modifiers in the Xcode library.

Performance analysis

Measure and improve your app's responsiveness.