Updates / WWDC23

WWDC23

Highlights of new technologies introduced at WWDC23.

Overview

Browse a selection of documentation for new technologies, frameworks, and APIs introduced at WWDC23. Many existing frameworks have added significant functionality, and you'll find new ways to enhance your apps targeting the latest platform release.

For a comprehensive list of downloadable sample code projects, see WWDC23 Sample Code. For the latest design guidance localized in multiple languages, see Human Interface Guidelines > What's New.

Topics

visionOS

visionOS is a brand new platform that allows you to build immersive apps and games for spatial computing that run on Apple Vision Pro. Create new apps using SwiftUI to take full advantage of the spectrum of immersion available in visionOS. If you have an existing iPad or iPhone app, add the visionOS destination to your app's target to gain access to the standard system appearance, and add platform-specific features to create a compelling experience.

visionOS
Create a new universe of apps and games for Apple Vision Pro.

Hello World
Use windows, volumes, and immersive spaces to teach people about the Earth.

Creating your first visionOS app

Build a new visionOS app using SwiftUI and add platform-specific features.

Designing for visionOS

When people wear Apple Vision Pro, they enter an infinite 3D space where they can engage with your app or game while staying connected to their surroundings.

Adding 3D content to your app

Add depth and dimension to your visionOS app and discover how to incorporate your app’s content into a person’s surroundings.

Bringing your existing apps to visionOS

Build a version of your iPadOS or iOS app using the visionOS SDK, and update your code for platform differences.

SwiftData

Use SwiftData with SwiftUI to create a seamless connection from your data model to user interface. Like SwiftUI, SwiftData focuses entirely on code, with no external file formats to manage. Instead, it uses Swift’s new macro system to offer a streamlined API. SwiftData uses the `Codable` protocol to understand structures and enumerations, so you can model your data with the tools you already know. These types are fully modeled in the underlying data store, enabling you to perform fast and efficient queries, even on complex structured data.

SwiftData

Write your model code declaratively to add managed persistence and efficient model fetching.

Building a document-based app using SwiftData

Code along with the WWDC presenter to transform an app with SwiftData.

Widgets, Live Activities, and watchOS complications

Bring your widgets to new places like the macOS desktop, Standby, and the Lock Screen on iPad. And now many of your widgets also gain new interactive abilities using `Button` and `Toggle`. Support Live Activities to keep users updated with the latest data from your app. Add animations in your widget to respond to user action or data changes.

WidgetKit

Extend the reach of your app by creating widgets, watch complications, Live Activities, and controls.

Developing a WidgetKit strategy

Explore features, tasks, related frameworks, and constraints as you make a plan to implement widgets, controls, watch complications, and Live Activities.

{} Emoji Rangers: Supporting Live Activities, interactivity, and animations

Offer Live Activities, controls, animate data updates, and add interactivity to widgets.

Creating a widget extension

Display your app’s content in a convenient, informative widget on various devices.

Making network requests in a widget extension

Update your widget with new information you fetch with a network request.

Creating views for widgets, Live Activities, and watch complications

Implement glanceable views with WidgetKit and SwiftUI.

Creating accessory widgets and watch complications

Support accessory widgets that appear on the Lock Screen and as complications on Apple Watch.

Supporting additional widget sizes

Offer widgets in additional contexts by adding support for various widget sizes.

Preparing widgets for additional platforms, contexts, and appearances

Create widgets that support additional platforms and adapt to their context.

Adding interactivity to widgets and Live Activities

Include buttons or toggles in a widget or Live Activity to offer app functionality without launching the app.

Animating data updates in widgets and Live Activities

Use SwiftUI animations to indicate data updates in your widgets and Live Activities.

Linking to specific app scenes from your widget or Live Activity

Add deep links to your widgets and Live Activities that enable people to open a specific scene in your app.

Making a configurable widget

Give people the option to customize their widgets by adding a custom app intent to your project.

Migrating widgets from SiriKit Intents to App Intents

Configure your widgets for backward compatibility.

Keeping a widget up to date

Plan your widget’s timeline to show timely, relevant information using dynamic views, and update the timeline when things change.

Increasing the visibility of widgets in Smart Stacks

Provide contextual information and donate intents to the system to make sure your widget appears prominently in Smart Stacks.

ActivityKit

Share live updates from your app as Live Activities on iPhone, iPad, Apple Watch, and the Mac.

Displaying live data with Live Activities

Display up-to-date data and offer quick interactions in the Dynamic Island, on the Lock Screen, in CarPlay, and on a paired Mac or Apple Watch.

SwiftUI

Use the ever-expanding SwiftUI API in your apps, with greater control over scroll and focus behavior, and more. Build sophisticated animations with advanced new capabilities, and even automatically match the speed of your animation to the velocity of user gestures. Share more SwiftUI code with your watchOS app using new `TabView`, `ToolbarItem`, and `NavigationSplitView`. And use `@Observable` with SwiftUI to automatically detect and update only the fields of your views that people access.

SwiftUI updates

Learn about important changes to SwiftUI.

Observation

Make responsive apps that update the presentation when underlying data changes.

Backyard Birds: Building an app with SwiftData and widgets

Create an app with persistent data, interactive widgets, and an all new in-app purchase experience.

```
struct ContentUnavailableView<Label, Description, Actions> where Label
: View, Description : View, Actions : View
```

An interface, consisting of a label and additional content, that you display when the content of your app is unavailable to users.

```
@preconcurrency protocol CustomAnimation : Hashable, Sendable
```

A type that defines how an animatable value changes over time.

```
struct PhaseAnimator<Phase, Content> where Phase : Equatable, Content :
View
```

A container that animates its content by automatically cycling through a collection of phases that you provide, each defining a discrete step within an animation.

```
struct TableColumnCustomization<RowValue> where RowValue : Identifiable
```

A representation of the state of the columns in a table.

```
struct OutlineGroup<Data, ID, Parent, Leaf, Subgroup> where Data :
RandomAccessCollection, ID : Hashable
```

A structure that computes views and disclosure groups on demand from an underlying collection of tree-structured, identified data.

```
@MainActor @preconcurrency struct SectorMark
```

A sector of a pie or donut chart, which shows how individual categories make up a meaningful total.

Xcode and developer tools

With the latest Xcode release, you can verify the origin of the frameworks your app depends upon, and add metadata to your own frameworks to ensure other developers about your privacy policies. Learn to install and use on-demand simulators. Enhance your Xcode Cloud build workflows to automate packaging up your app for distribution. And the new strings catalog feature right within Xcode gives even more control over how your UI text is handled in locales around the globe.

Xcode updates

Learn about important changes to Xcode.

Downloading and installing additional Xcode components

Add more Simulator runtimes, optional features, and support for additional platforms.

Localizing and varying text with a string catalog

Use a string catalog to translate text, handle plurals, and vary the text your app displays on specific devices.

Capabilities

Enable services that Apple provides, such as In-App Purchase, Push Notifications, Apple Pay, iCloud, and many others.

Verifying the origin of your XCFrameworks

Discover who signed a framework, and take action when that changes.

Configuring your project to use mergeable libraries

Use mergeable dynamic libraries to get app launch times similar to static linking in release builds, without losing dynamically linked build times in debug builds.

Describing data use in privacy manifests

Declare the data collected by your app or by third-party SDKs.

Distributing your app for beta testing and releases

Release your app to beta testers and users.

Creating a workflow that builds your app for distribution

Configure a workflow to build and sign your app for distribution to testers with TestFlight, in the App Store, or as a notarized app.

Debugging

Identify and address issues in your app using the Xcode debugger, Xcode Organizer, Metal debugger, and Instruments.

watchOS

Add new capabilities to your watchOS apps, and update your interface to correspond to the latest interface guidance for watchOS 10. Adopt WidgetKit features with Apple Watch-specific experiences, or use WorkoutKit to build apps that support better health.

watchOS updates

Learn about important changes to watchOS.

Updating your app and widgets for watchOS 10

Integrate SwiftUI elements and watch-specific features, and build widgets for the Smart Stack.

Designing for watchOS

When people glance at their Apple Watch, they know they can access essential information and perform simple, timely tasks whether they’re stationary or in motion.

Increasing the visibility of widgets in Smart Stacks

Provide contextual information and donate intents to the system to make sure your widget appears prominently in Smart Stacks.

WorkoutKit

Create, preview, and sync workout compositions to the Workout app.

Messages apps and stickers

Give your Messages app and stickers more power than ever before, with stickers available from the keyboard picker in apps all across the system.

Messages

Create app extensions that allow users to send text, stickers, media files, and interactive messages.

Adding Sticker packs and iMessage apps to the system Stickers app, Messages camera, and FaceTime

Enable your Sticker pack or iMessage app in the media context.

`enum MSMessagesAppPresentationContext`

Presentation contexts describing where your iMessage app appears.

UIKit

Simplify spring animations by providing duration and bounce parameters for the new view method, `animate`. Take advantage of other new UI controls and behaviors, including improvements to the presentation of `UIStatusBar` using the new default option.

UIKit updates

Learn about important changes to UIKit.

`CFBundleDocumentTypes`

The document types supported by the bundle.

```
@MainActor @preconcurrency class func animate(springDuration duration: TimeInterval = 0.5, bounce: CGFloat = 0.0, initialSpringVelocity: CGFloat = 0.0, delay: TimeInterval = 0.0, options: UIView.Animation Options = [], animations: () -> Void, completion: ((Bool) -> Void)? = nil)
```

Animates changes to one or more views using a spring animation with the specified duration, bounce, initial velocity, delay, options, and completion handler.

```
@MainActor func viewIsAppearing(_ animated: Bool)
```

Notifies the view controller that the system is adding the view controller’s view to a view hierarchy.

`struct UIContentUnavailableConfiguration`

A content configuration for a content-unavailable view.

`CFBundleDocumentTypes`

The document types supported by the bundle.

`@MainActor var allowsKeyboardScrolling: Bool { get set }`

A Boolean value that determines whether the scroll view allows scrolling its content with hardware keyboard input.

``case `default` ``

A style that automatically selects an appearance for the status bar and updates it dynamically to maintain contrast with the content below it.

Audio, video, and media

Build entirely new Apple TV experiences with access to the Continuity Camera, enabling video conferencing and other types of apps on the biggest screen in your home or office. Use the Cinematic framework to add support for editing movies filmed in Cinematic mode from the Camera app.

{} Supporting Continuity Camera in your tvOS app

Capture high-quality photos, video, and audio in your Apple TV app by connecting an iPhone or iPad as a continuity device.

Cinematic

Integrate playback and editing of assets captured in Cinematic mode into your app.

SensitiveContentAnalysis

Provide a safer experience in your app by detecting and alerting users to nudity in images and videos before displaying them onscreen.

`@MainActor class AVContinuityDevicePickerViewController`

A view controller that provides an interface to a person so they can select and connect a continuity device to the system.

Metal

With Metal debugging and performance analysis tools, you can make your apps and games perform their best.

Metal debugger

Debug and profile your Metal workload with a GPU trace.

Metal developer workflows

Locate and fix issues related to your app’s use of the Metal API and GPU functions.

MetalFX

Boost your Metal app’s performance by upscaling lower-resolution content to save GPU time.

Maps and location

MapKit for SwiftUI

MapKit for SwiftUI allows you to build map-centric views and apps across Apple platforms. You can design expressive and highly interactive Maps with minimal code by composing views, using ViewBuilders and view modifiers.

{} Monitoring location changes with Core Location

Define boundaries and act on user location updates.

Core Location

Obtain the geographic location and orientation of a device.

App Store and distribution

StoreKit

Support In-App Purchases and interactions with the App Store.

App Store Server API

Manage your customers’ App Store transactions from your server.

App Store Server Notifications changelog

Learn about changes to the App Store Server Notifications service.

App Store Connect API Release Notes

Learn about new features and updates in the App Store Connect API.

Security and privacy

Improve your app and website security, while protecting your user’s privacy, using the latest SDK features. Autofill password fields to easily employ passkeys, as well as saved passwords. And interact with the user’s calendar store using the `EKEventStore` API.

`@MainActor class ASCredentialProviderViewController`

A view controller that a credential manager app uses to extend AutoFill.

Accessing files from the macOS App Sandbox

Read and write documents and supporting files while maintaining security protection.

Accessing the event store

Request access to a person’s calendar data through the event store.

`enum HPKE`

A container for hybrid public key encryption (HPKE) operations.

Extensions and XPC

Use ExtensionKit for macOS, and now for your iOS and iPadOS apps, to create extensions that expose a secure method for other apps to interact and extend your app. And a new Swift-specific API for XPC can make your code even easier to manage.

XPC updates

Learn about important changes to XPC.

`class XPCListener`

A type that performs tasks for clients across process boundaries.

`class XPCSession`

A type that sends messages to a server process.

ExtensionKit

Make custom UI from an app extension available in a host app, and manage the list of enabled and disabled app extensions.

ExtensionFoundation

Create executable bundles to extend the functionality of other apps.

Group activities and sharing

Use a `GroupSessionJournal` object to transfer files and other data objects between participants of a shared activity.

`final class GroupSessionJournal`

An object that manages file and data transfers between participants joined in a group session.

{} Drawing content in a group session

Invite your friends to draw on a shared canvas while on a FaceTime call.

Group Activities

Create app-specific activities your users can share and experience together.

Machine learning

Creating an Image Classifier Model

Train a machine learning model to classify images, and add it to your Core ML app.

VisionKit

Identify and extract information in the environment using the device’s camera, or in images that your app displays.

Health

Use HealthKit to securely and privately store user health data on their device, with new support for iPadOS.

HealthKit updates

Learn about important changes to HealthKit.

Apple Pay and Wallet

ProximityReader

Read contactless physical and digital wallet cards using your iPhone.

{} Checking IDs with the Verifier API

Read and verify mobile driver’s license information without any additional hardware.

```
@MainActor @preconcurrency struct PayLaterView<FallbackView> where
FallbackView : View
```

A view that displays the Apple Pay Later visual merchandising widget.

Hardware and virtual machines

SensorKit

Retrieve data and derived metrics from sensors on an iPhone, or paired Apple Watch.

DockKit

Interact with accessories that track subjects on camera as they move around.

Virtualization

Create virtual machines and run macOS and Linux-based operating systems.

Screen capture

ScreenCaptureKit updates

Learn about important changes to ScreenCaptureKit.

Symbols

Symbols

Apply universal animations to symbol-based images.

See Also

WWDC

WWDC25

Highlights of new technologies introduced at WWDC25.

WWDC24

Highlights of new technologies introduced at WWDC24.

WWDC22

Highlights of new technologies introduced at WWDC22.

WWDC21

Highlights of new technologies introduced at WWDC21.