Develop in Swift Tutorials

Educator Guide

Develop in Swift Tutorials introduce app development with Swift and Xcode to anyone learning to build apps for Apple platforms.

In each chapter, learners will complete:

Tutorials

- Coding a project, ranging from an app prototype to a fully functioning app
- Building on prior knowledge, getting progressively more challenging

Articles

- Review of concepts
- Ideas for extending an app
- Suggestions for how to apply skills in a different context, often by creating a new project

Choose your approach:

You can present the content linearly, or you can incorporate other text, documentation, tutorials, videos, and projects to fit your needs. One option is to have learners complete the tutorial independently, then choose items from the "Continue practicing" section to complete together, allowing learners to work collaboratively and ask questions.

App design

Plan, prototype, and design your app.

Discovery

Ask questions, listen to your users, and define a list of features.

Topics and skills

- App design cycle
- Defining goals
- Low-fidelity sketches
- Planning navigation
- Prioritizing features

Prototypes

Create an interactive prototype that looks and feels like a fully developed iOS app and is ready for testing.

Topics and skills

- High-fidelity sketches
- Modals
- Prototypes
- SF Pro font
- SF Symbols
- Tab bars
- Toolbars

Testing and validation

Improve the clarity and usability of your app design by observing how people interact with your prototype.

Topics and skills

- Identifying conclusions and root causes
- Selecting features for iteration
- Task flows
- Test scripts
- User testing

Iteration

Revisit and strengthen your app design using insights from testing and validation.

Topics and skills

- Accessibility
- Built-in views vs. custom components
- Color
- Content presentation
- Dark Mode
- Dynamic Type
- Typography
- Visual consistency

SwiftUI

Get familiar with some of the tools and technologies you'll use to create apps.

Explore Xcode

Get to know Xcode and SwiftUI by creating a prototype of a messaging app. Learn about syntax for Swift and how to use the source editor and preview.

Topics and skills

- Background
- **`Color`**
- Creating a new project
- Dot notation
- Modifiers
- Padding
- **`String`**
- Swift syntax
- **`Text`**
- Views
- Xcode error messages
- Xcode Library

Views, structures, and properties

Learn how to build a custom view to create a multiday weather forecast. In your view, you'll use properties to customize the display for each day.

Topics and skills

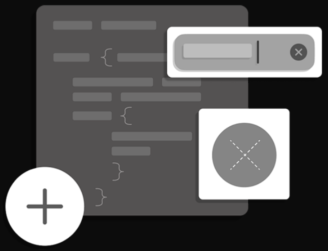
- Arguments and parameters
- **`Bool`**
- Computed properties
- Custom subviews
- **`Font`**
- Foreground style
- **`Image`**
- Initializers
- **`Int`**
- **`HStack`** and **`VStack`**
- Returning a value
- SF Symbols
- Stored properties
- String interpolation
- Structures
- Subviews
- Type annotation

Layout and style

Build two onboarding screens for an iOS app to learn useful tools for putting views where you want them onscreen and inspecting their size. Define new colors in the asset catalog and use them to create gradient backgrounds.

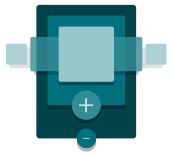
Topics and skills

- Accent color
- Arrays
- Borders
- Brightness
- Color assets
- Customizing a preview
- **`Font`**
- Frames
- **`Gradient`**
- **`Image`**
- Pinning a preview
- **`Shape`**
- **`Spacer`**
- **`TabView`**
- Transparency
- Type inference
- **`ZStack`**



SwiftUI

Get familiar with some of the tools and technologies you'll use to create apps.

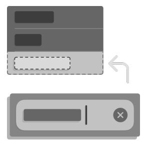


Buttons and state

Explore adding buttons to your apps. Learn about Swift closures and their relationship to buttons. Use state properties to update the user interface automatically.

Topics and skills

- Animation
- Aspect ratio
- Assignment operator
- **`Button`**
- Button styles
- Closures
- **`Color`**
- Disabling controls
- Dynamic sizing
- Equality operator
- **`ForEach`**
- Hierarchical SF Symbols
- Randomization
- Range operator
- Resizable images
- **`@State`**
- Trailing closure syntax
- View tint



Lists and text fields

Create a dynamic interface that stores a set of items in an array and displays them using lists. Use text fields and bindings to let people enter text.

Topics and skills

- Arrays
- Adding and removing from arrays
- Bindings
- Buttons with custom labels
- Disabling autocorrection
- Clip shapes
- **`ForEach`**
- **`List`**
- Not (!) operator
- Symbol rendering modes
- Ternary conditional operator
- **`TextField`**
- **`Toggle`**

Data modeling

Model real-world concepts and relationships by creating and testing your own custom types.

Custom types and Swift Testing

Define your first data model by making your own custom types, and prove they work correctly with unit tests. Then use your custom types to keep track of scores in a game.

Topics and skills

- Creating a type to contain your app's logic
- Creating **`enum`** types
- Creating **`struct`** types
- Creating unit tests
- Fixing test failures
- **`Grid`** and **`GridRow`**
- **`Identifiable`** and **`UUID`**
- **`.opacity`** and **`.disabled`**
- Running tests
- Swift file creation

Models and persistence

Build a list of your friends' birthdays, using SwiftData to save and retrieve that data across launches.

Topics and skills

- **`Calendar`**
- Classes
- Data models
- **`Date`**
- Date formatting
- **`DatePicker`**
- **`@Environment`**
- Frameworks
- **`@Model`** macro
- **`NavigationStack`**
- **`@Query`** macro
- Safe area
- SwiftData context

Navigation, editing, and relationships

Create an app to track friends and their favorite movies using SwiftData to manage the model objects. Use a query to display the items in a list, and make a detail view to edit them. Then learn how to create and display relationships between friends and movies, and explore how to create advanced queries.

Topics and skills

- **`@Bindable`**
- **`ContentUnavailableView`**
- Creating sample data
- Custom view initializers
- Environment **`dismiss`** value
- **`Form`**
- **`Group`**
- Modal interfaces
- Multiple previews
- **`ModelConfiguration`** **`ModelContainer`**
- Model relationships
- Navigation hierarchies
- **`NavigationLink`**
- **`NavigationSplitView`**
- Or (**`||`**) operator
- **`Picker`**
- Predicate
- Property wrappers
- Refactoring
- **`Schema`**
- Search
- **`Section`**
- Sheets
- Sorting arrays
- Toolbars
- View tags

Observation and shareable data models

Power an alphabet game using Observation. Share a complex data model with many independent views.

Topics and skills

- **`Dictionary`**
- Documentation comments
- **`@Observable`**
- **`onChange`**
- Sharing your types through the environment
- **`Task.sleep`**
- Xcode's Quick Help and jump bar
- **`zip`**

App development

Build and debug a fully functioning habit-tracking app. Apply coding fundamentals and accessibility principles.

Views and data storage

Build a screen to capture text and photo data. Use SwiftData to save entries and build a custom view to display them.

Topics and skills

- **`confirmationDialog`**
- Laying out and modifying SwiftUI components
- **`Mask`**
- **`PhotosPicker/PhotoKit`**
- **`scaledToFill()`**
- **`Transferable`**

User experience features

Use your knowledge of Swift and data modeling to make the app more engaging with user experience features.

Topics and skills

- Creating and reusing SwiftUI views
- Creating, updating, and querying using SwiftData
- **`FetchDescriptor`**
- **`reversed`**
- **`scrollTransition`**
- **`textSelection(.enabled)`**

App refinement

Add localization and accessibility features to your app. Then debug the app so it's ready to test.

Topics and skills

- **`AttributedString`**
- Dark Mode
- defaultScrollAnchor
- Dynamic Type
- **`fixedSize`**
- Locale
- **`scrollBounceBehavior()`**
- **`print`**
- Xcode console

Machine learning

Enhance your apps with machine learning.

Natural language

Build a sentiment analysis app and use the Natural Language framework to analyze responses to an open-ended survey prompt.

Topics and skills

- **`@FocusState`** wrapper
- **`Chart`**
- **`chartProxy`**
- Charts framework
- **`GeometryReader`**
- **`insert`** versus **`append`**
- Natural Language framework
- **`NLTagger`**
- **`Plottable`** protocol
- **`ScrollView`**
- Sentiment analysis
- **`Textfield.axis`**

Recognize text in images

Create an app that uses the Vision Framework and the Translation API to translate text on signs.

Topics and skills

- Alert for app processing time
- **`ImageResource`**
- Overlays
- **`RecognizedTextObservation`** and **`RecognizeTextRequest`**
- **`Shape`**
- **`Translation framework`**
- **`.translationPresentation`**
- **`ViewModifier`** protocol
- Vision framework

Model training with Create ML

Use Xcode's Create ML tool to train a model to estimate the anticipated difficulty of a hike using provided data.

Topics and skills

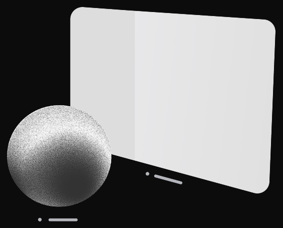
- Create ML tool in Xcode
- CSV files
- Machine learning algorithms
- Model accuracy
- Previewing output
- Training, validation, and testing data
- Xcode developer tools

Custom models with Core ML

Integrate a custom machine learning model into an app that predicts the difficulty of an upcoming hike.

Topics and skills

- Adding a Core ML model to an app
- **`CaseIterable`** protocol
- Core ML framework
- Generic views
- Segmented pickers
- View builders



Spatial computing

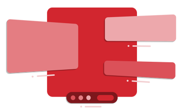
Design app experiences for spatial computing.

Windows in visionOS

Create your first visionOS app with a window using SwiftUI.

Topics and skills

- **`Circle`**
- **`ColorPicker`**
- **`Double`**
- **`Grid`**
- **`GridRow`**
- Padding for 3D views
- Remainder (%) operator
- **`Slider`**
- visionOS simulator
- Window resizability
- Windows

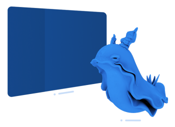


Ornaments and multiple windows

Create multiple windows in visionOS using SwiftUI. Use ornaments to provide access to frequently used controls without crowding or obscuring window contents.

Topics and skills

- **`@Environment isEnabled`**
- **`@Environment openWindow`**
- **`.glassBackgroundEffect`**
- **`@Previewable`** previews
- **`TextField`** word wrapping
- visionOS **`.ornament`**
- **`WindowGroup`**, **`.windowStyle`**, and **`.windowResizability`**



Volumes in visionOS

View 3D content from any angle in the Shared Space using Reality Composer Pro and SwiftUI.

Topics and skills

- Arrays
- **`DragGesture`**
- Environment **`openWindow`** value
- **`Model3D`**
- **`NavigationSplitView`**
- Reality Composer Pro
- Rotation in three dimensions
- Toolbars
- Volumes
- **`WindowGroup`**

App distribution

Use TestFlight to test your beta app, get feedback, and make improvements. Learn how to prepare your app for publishing when it's ready.

Preparation for distribution

Collect the necessary information to get your project ready for testing and publishing.

Topics and skills

- App category
- App icon
- Bundle ID
- Supported devices and SDKs
- Supported orientations
- Version and build numbers

Testing and feedback

Test your beta app using TestFlight to receive feedback.

Topics and skills

- Creating and uploading an archive
- Creating a tester group in App Store Connect
- Sharing your beta app with testers using TestFlight
- Responding to feedback
- Incrementing your build and uploading a newer version

Review and distribution

Get your app ready for review and publishing.

Topics and skills

- App metadata
- App review
- Common review issues
- Product page
- Submitting to the App Store