Framework

Swift

Build apps using a powerful open language.

iOS 8.0+ | iPadOS 8.0+ | Mac Catalyst 13.0+ | macOS 10.10+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

Overview

Swift includes modern features like type inference, optionals, and closures, which make the syntax concise yet expressive. Swift ensures your code is fast and efficient, while its memory safety and native error handling make the language safe by design. Writing Swift code is interactive and fun in Swift Playgrounds, playgrounds in Xcode, and REPL.

```
var interestingNumbers = [
    "primes": [2, 3, 5, 7, 11, 13, 17],
    "triangular": [1, 3, 6, 10, 15, 21, 28],
    "hexagonal": [1, 6, 15, 28, 45, 66, 91]
]

for key in interestingNumbers.keys {
    interestingNumbers[key]?.sort(by: >)
}

print(interestingNumbers["primes"]!)
// Prints "[17, 13, 11, 7, 5, 3, 2]"
```

Learn Swift

If you're new to Swift, read The Swift Programming Language for a quick tour, a comprehensive language guide, and a full reference manual. If you're new to programming, check out Swift Playgrounds on iPad.

Swift is developed in the open. To learn more about the open source Swift project and community, visit Swift.org.

Topics

Essentials

Swift updates

Learn about important changes to Swift.

Adopting strict concurrency in Swift 6 apps

Enable strict concurrency checking to find data races at compile time.

Standard Library

`struct Int`

A signed integer value type.

`struct Double`

A double-precision, floating-point value type.

`struct String`

A Unicode string value that is a collection of characters.

`struct Array`

An ordered, random-access collection.

`struct Dictionary`

A collection whose elements are key-value pairs.

Swift Standard Library

Solve complex problems and write high-performance, readable code.

Observation

Observation

Distributed Actors

Distributed

Build systems that run distributed code across multiple processes and devices.

Regular Expression DSL

RegexBuilder

Use an expressive domain-specific language to build regular expressions, for operations like searching and replacing in text.

Low-Level Atomic Operations

Synchronization

Build synchronization constructs using low-level, primitive operations.

Data Modeling

Choosing Between Structures and Classes

Decide how to store data and model behavior.

Adopting Common Protocols

Make your custom types easier to use by ensuring that they conform to Swift protocols.

Data Flow and Control Flow

Maintaining State in Your Apps

Use enumerations to capture and track the state of your app.

Preventing Timing Problems When Using Closures

Understand how different API calls to your closures can affect your app.

Language Interoperability with Objective-C and C

Objective-C and C Code Customization

Apply macros to your Objective-C APIs to customize how they're imported into Swift.

Migrating Your Objective-C Code to Swift

Learn the recommended steps to migrate your code.

Cocoa Design Patterns

Adopt and interoperate with Cocoa design patterns in your Swift apps.

Handling Dynamically Typed Methods and Objects in Swift

Cast instances of the Objective-C `id` type to a specific Swift type.

Using Objective-C Runtime Features in Swift

Use selectors and key paths to interact with dynamic Objective-C APIs.

Imported C and Objective-C APIs

Use native Swift syntax to interoperate with types and functions in C and Objective-C.

Calling Objective-C APIs Asynchronously

Learn how functions and methods that take a completion handler are converted to Swift asynchronous functions.

Language Interoperability with C++

{} Mixing Languages in an Xcode project

Use C++ APIs in Swift – and Swift APIs in C++ – in a single framework target, and consume the framework’s APIs in a separate app target.

{} Calling APIs Across Language Boundaries

Use a variety of C++ APIs in Swift – and vice-versa – across multiple targets and frameworks in an Xcode project.

Structures

`struct OutputRawSpan`