Article

Adopting strict concurrency in Swift 6 apps

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

Overview

Strict concurrency checking in the Swift 6 language mode helps you find and fix data races at compile time.

Overlapping access to shared mutable state creates the risk for data races. Data races can cause your app to crash, misbehave, or corrupt user data. Because data races depend on the ordering of concurrent operations, they can be very difficult to reproduce and debug. Strict concurrency checking lets you confirm, when you compile your app, that its code is free from data races. When you identify a data race, you fix it by eliminating overlapping access, shared access, or mutable state.

For information about the language-level concurrency model in Swift, see Concurrency in The Swift Programming Language.

Decide when to upgrade to strict checking

Xcode 16 supports the new Swift 6 language mode, which uses strict checking for concurrency, and it also still includes the Swift 5, 4.2, and 4 language modes. The Swift 6 language mode is opt-in: Your projects continue to build with their current language mode, and new projects default to the Swift 5 language mode.

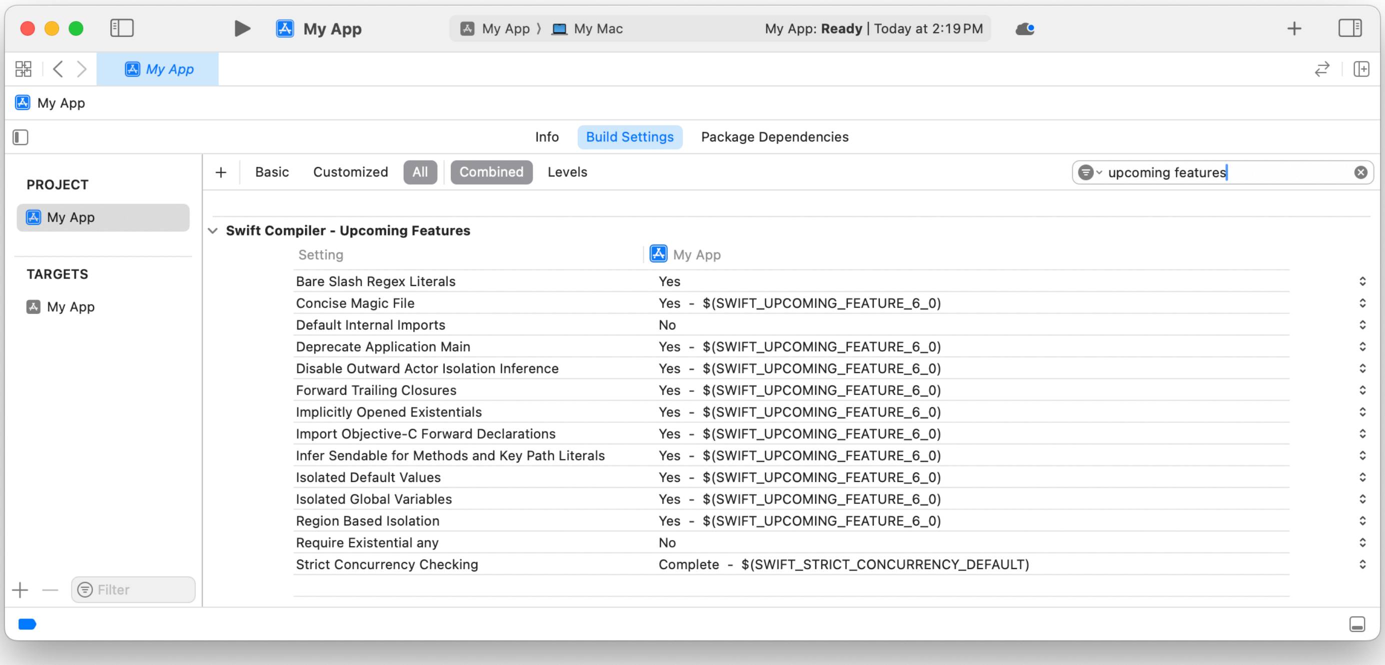
Adopting the Swift 6 language mode can significantly improve the quality of your app by catching mistakes in concurrent code at compile time. It can be especially useful if you’re experiencing hard-to-reproduce crashes and want to methodically eliminate the risk of data races. If you’re actively working on integrating more concurrency to improve responsiveness and performance, adopting the Swift 6 language mode can ensure that those changes don't risk introducing new data races.

If your app is organized into multiple modules, you can migrate your code one module at a time. It's usually easier to migrate the app first, and the migrate the modules it depends on.

If you maintain a public Swift package, adopting the Swift 6 language mode helps your users who want to migrate their codebases too. You can follow along on the adoption of the Swift 6 language in popular packages on `SwiftPackageIndex.com`.

Upgrade a project to Swift 6

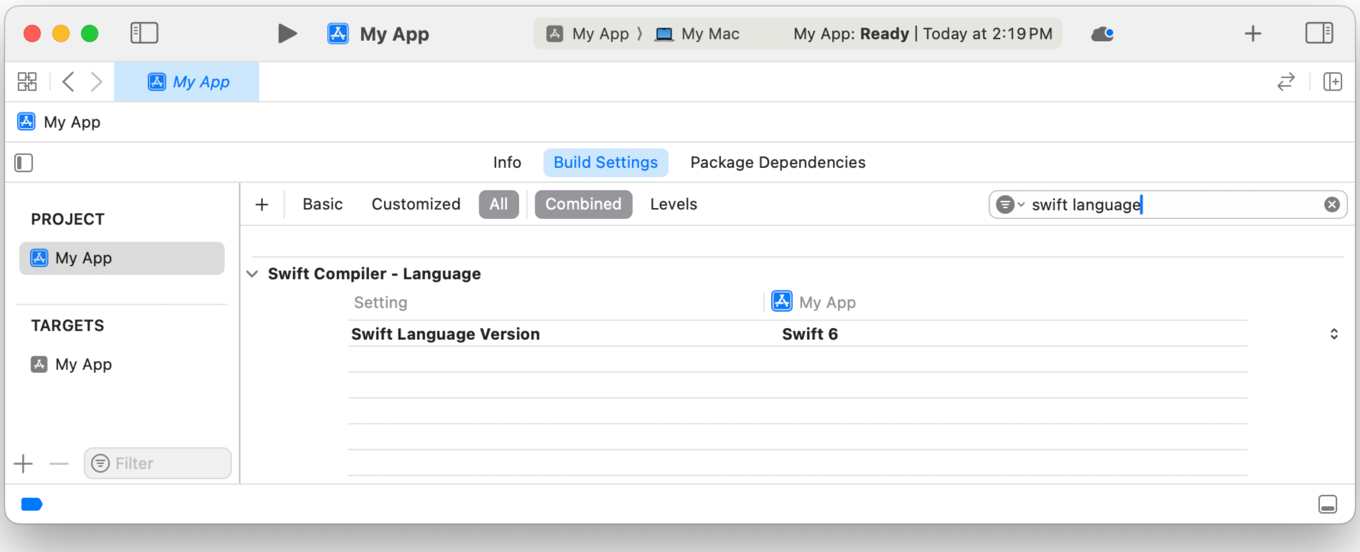
You can start using new features from Swift 6, like strict concurrency checking, before upgrading your project to the Swift 6 language mode. To enable new language features, open your build settings, and choose Swift Compiler - Upcoming Features; then select Yes next to the features you want to adopt. As you prepare to upgrade to the Swift 6 language version, you can increase the level of concurrency checking by changing the Strict Concurrency Checking build setting from Minimal to Complete.



Swift Compiler - Upcoming Features

Setting	My App
Bare Slash Regex Literals	Yes
Concise Magic File	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Default Internal Imports	No
Deprecate Application Main	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Disable Outward Actor Isolation Inference	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Forward Trailing Closures	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Implicitly Opened Existentials	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Import Objective-C Forward Declarations	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Infer Sendable for Methods and Key Path Literals	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Isolated Default Values	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Isolated Global Variables	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Region Based Isolation	Yes - $(SWIFT_UPCOMING_FEATURE_6_0)
Require Existential any	No
Strict Concurrency Checking	Complete - $(SWIFT_STRICT_CONCURRENCY_DEFAULT)

To upgrade your project to the Swift 6 language mode, open your build settings, and choose Swift Compiler - Language > Swift Language Version.



Swift Compiler - Language

Setting	My App
Swift Language Version	Swift 6

Upgrading to the Swift 6 language mode enables all of the new language features that are part of Swift 6, including strict concurrency checking.

> **Note**
>
> After you upgrade, the list of upcoming features still includes a few features that will be part of a future Swift version, which are disabled by default in the Swift 6 language mode.

After you turn on strict concurrency checking, you might see a large number of new compiler errors and warnings. However, fixing the data race in one part of your app often resolves multiple reported errors. You don't have to fix all of the reported data races at once. If you need to, you can fix some of them now, switch back to the Swift 5 language mode or switch back to minimal concurrency checking, and then continue fixing the remaining data races later.

For information about how to migrate, including techniques for interoperating with code that hasn't yet migrated to strict checking, see Migrating to Swift 6.

See Also

Essentials

Swift updates

Learn about important changes to Swift.