BACK TO THE FUTURE

SWIFT 6 EDITION

Hi! I’m Pol

App maker and content creator based in Barcelona


Glovo


ANNOUNCEMENT

HELM IS SHIPPING TODAY!

When Apple approve it!

POLPIELLA.DEV


RUNWAY
runway.team


Discovery de la NASA
NASA
SPACE SHUTTLE
Discovery
Discovery
USA
LEGO
18+ | 10283 | 2354 pcs/pzs

HOW TO GET READY FOR

SWIFT 6

Swift 5 Released!

MARCH 25, 2019

 Ted Kremenek

Swift 5 is now officially released!

Swift 5 is a major milestone in the evolution of the language. Thanks to ABI stability, the Swift runtime is now included in current and future versions of Apple's platform operating systems: macOS, iOS, tvOS and watchOS. Swift 5 also introduces new capabilities that are building blocks for future versions, including a reimplementation of String, enforcement of exclusive access to memory during runtime, new data types, and support for dynamically callable types.

You can try out some of the new features in this playground put together by Paul Hudson.

Language Updates

Stable ABI and Binary Compatibility

The ABI is now declared stable for Swift 5 on Apple platforms. As a result, the Swift libraries are now incorporated into every macOS, iOS, tvOS, and watchOS release going forward. Your apps will be easier to build and smaller because they won't have to include those libraries.

See these blog posts for more details:

- ABI Stability and More
- Evolving Swift On Apple Platforms After ABI Stability


MAJOR VERSION
BREAKING CHANGES
PATCH VERSION
BUG FIXES, ETC.
6.0.0
MINOR VERSION
NON-BREAKING CONSIDERABLE CHANGES


MAJOR VERSION
BREAKING CHANGES
PATCH VERSION
BUG FIXES, ETC.
6.0.0
MINOR VERSION
NON-BREAKING CONSIDERABLE CHANGES

SE-0412 SE-0364 SE-0418 SE-0411

SE-0337 SE-0352 SE-0421

SE-0270 SE-0274 SE-0410 SE-0416

SE-0408

SE-0426

SWIFT 6

SE-0286

SE-0383 SE-0409 SE-0384 SE-0220

SE-0401 SE-0414 SE-0354

SE-0301 SE-0405 SE-0420 SE-0422

SE-0412 SE-0364 SE-0418 SE-0411

SE-0337 SE-0352 SE-0421

SE-0270 SE-0274 SE-0410 SE-0416

SE-0408

SE-0426

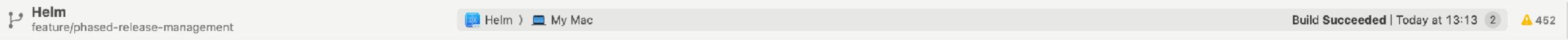
SWIFT 6

SE-0286

SE-0383 SE-0409 SE-0384 SE-0220

SE-0401 SE-0414 SE-0354

SE-0301 SE-0405 SE-0420 SE-0422


Helm
feature/phased-release-management
Helm 〉 My Mac
Build Succeeded | Today at 13:13
2
452

Experience

Goose farmer
Self-employed
Jul 2023 - Present · 11 mos
On-site

Microsoft
22 yrs 4 mos

Principal Performance Architect
Full-time
Jul 2022 - Jul 2023 · 1 yr 1 mo
Chehalis, Washington, United States

Principal Software Development Engineer
Apr 2001 - Jul 2022 · 21 yrs 4 mos
Redmond,WA

Azure Performance Team, August 2017 - Present
Office 365 Foundation Engineering...

...see more

Hewlett Packard
7 yrs 10 mos

Senior Software Engineer
1996 - 2001 · 5 yrs

Printer drivers. Wrote the GDI book then.

Senior Software Engineer
Apr 1993 - Jun 1996 · 3 yrs 3 mos

APCD (notebook, palmtop, calculater, WinCE)

Language Mode!

SWIFT 6

Holly Borla
@hollyborla

Let's talk about data race safety in Swift 6!

The Swift 6 language mode is opt-in; you can continue to use Swift 5 mode with the Swift 6 compiler. You can incrementally refine your code to take advantage of full data isolation at your own pace, module by module.

2:11 AM · Feb 16, 2024 · **57.1K** Views

6 56 217 69

Related posts

Post your reply

Reply

Holly Borla @hollyborla · Feb 16

Swift 6 code can depend on libraries using Swift 5 and vice versa.

If you're not familiar with Swift concurrency and data isolation, that's okay! There's no rush, and the language mode will be there when you're ready to take it on.

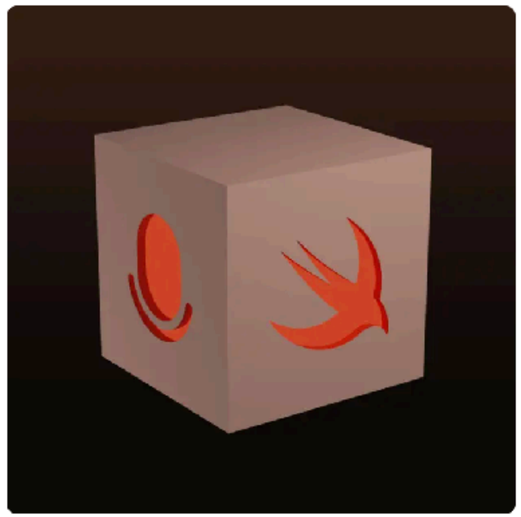
1 6 48 8.2K

Holly Borla @hollyborla · Feb 16

(I still don't post here, but I know a lot of Swift programmers do, and this is an important message to spread)

1 1 37 4.6K

Apple Podcasts Preview



1 hr 6 min PLAY

43: Now I'm worried our metrics aren't correct! with special guest Holly Borla

Swift Package Indexing

Technology

Listen on Apple Podcasts ↗

This week we had the opportunity to talk to Holly Borla who manages the Swift Compiler Team at Apple. We chat about upcoming Swift 6 changes and why they're a big deal, but also why you shouldn't worry too much. Of course, all three of us pick packages, too!
Interview with Holly
SE-0414: Region isolation SE-0431: Dynamically isolated function typesPackages
GRDB by Gwendal RouéGwendal's forum thread about adding Sendable annotationsPack by Matt CoxGeoURI by Jeff JohnstonConcurrencyRecipes by Matt Massicottegenerative-ai-swift by GoogleFit by Oleh Korchytskyi

Episode Website ↗

More Episodes

HTTPS://PODCASTS.APPLE.COM/US/PODCAST/43-NOW-IM-WORRIED-OUR-METRICS-ARENT-CORRECT-WITH/ID1654567329?I=1000653584114

SE-0412 SE-0364 SE-0418 SE-0411

SE-0337 SE-0352 SE-0421

SE-0270 SE-0274 SE-0410 SE-0416

SE-0408

SE-0426

SWIFT 6

SE-0286

SE-0383 SE-0409 SE-0384 SE-0220

SE-0401 SE-0414 SE-0354

SE-0301 SE-0405 SE-0420 SE-0422

SE-0412 SE-0364 SE-0418 SE-0411

SE-0337 SE-0352 SE-0421

SE-0270 SE-0274 SE-0410 SE-0416

Without the Swift 6 language mode

SWIFT 6

SE-0408 SE-0286

SE-0426

SE-0383 SE-0409 SE-0384 SE-0220

SE-0401 SE-0414 SE-0354

SE-0301 SE-0405 SE-0420 SE-0422


I THINK THAT'S EVERYTHING,

HOW TO GET READY FOR THE SWIFT 6 LANGUAGE MODE

DOWNLOADING SWIFT 6

Swift 6.0 Release Process

Development Announcements

mishal_shah 2 Feb 23

Swift 6.0 Release Process

This post describes the release process, and estimated schedule for Swift 6.0.

Snapshots of Swift 6.0

Downloadable snapshots of the Swift 6.0 release branch will be posted regularly as part of continuous integration 187 testing. As support is available, snapshot downloads will be added for newly supported platforms.

Once Swift 6.0 is released, the official final builds will also be posted in addition to the snapshots.

Getting Changes into Swift 6.0

On **March 15th 2024** the `release/6.0` branch will be cut from `main` in the swift repository and related project repositories. This will contain the changes that will be released in Swift 6.0. After the branch is cut, changes can be landed on the branch via pull request if they meet the criteria for the release.

Philosophy on Taking Changes into Swift 6.0

- All language and API changes for Swift 6.0 will go through the Swift Evolution 581 process. Evolution proposals should aim to be completed by the branch date in order to increase their chances of impacting the Swift 6.0 release. Exceptions will be considered on a case-by-case basis, particularly if they tie in with the core goal of the release.
- Other changes (e.g., bug fixes, diagnostic improvements, SourceKit interface improvements) will be accepted based on their risk and impact.
- Low-risk test tweaks will also be accepted late into the release branch if it aids in the qualification

Swift 6.0 Development

Swift 6.0 snapshots are prebuilt binaries that are automatically created from `release/6.0` branch. These snapshots are not official releases. They have gone through automated unit testing, but they have not gone through the full testing that is performed for official releases.

Download	Date	Architecture	Docker Tag
Xcode	May 14, 2024	Universal Debugging Symbols	Unavailable
Ubuntu 20.04	May 14, 2024	x86_64 Signature (x86_64) aarch64 Signature (aarch64)	nightly-6.0-focal
Ubuntu 22.04	May 14, 2024	x86_64 Signature (x86_64) aarch64 Signature (aarch64)	nightly-6.0-jammy
CentOS 7	May 14, 2024	x86_64 Signature (x86_64)	nightly-6.0-centos7
Amazon Linux 2	May 14, 2024	x86_64 Signature (x86_64) aarch64 Signature (aarch64)	nightly-6.0-amazonlinux2
Red Hat Universal Base Image 9	May 14, 2024	x86_64 Signature (x86_64) aarch64 Signature (aarch64)	nightly-6.0-rhel-ubi9
Windows 10	April 30, 2024	x86_64	Unavailable

INSTALL ON MACOS

```
#!/bin/sh

# Install `swiftenv`
brew install kylef/formulae/swiftenv
echo 'if which swiftenv > /dev/null; then eval "$(swiftenv init -)"; fi' >> ~/.zshrc
```

INSTALL ON MACOS

```
#!/bin/sh

# Install `swiftenv`
brew install kylef/formulae/swiftenv
echo 'if which swiftenv > /dev/null; then eval "$(swiftenv init -)"; fi' >> ~/.zshrc

# Install the latest Swift 6 development toolchain
swiftenv install 6.0-DEVELOPMENT-SNAPSHOT-2024-04-30-a
```

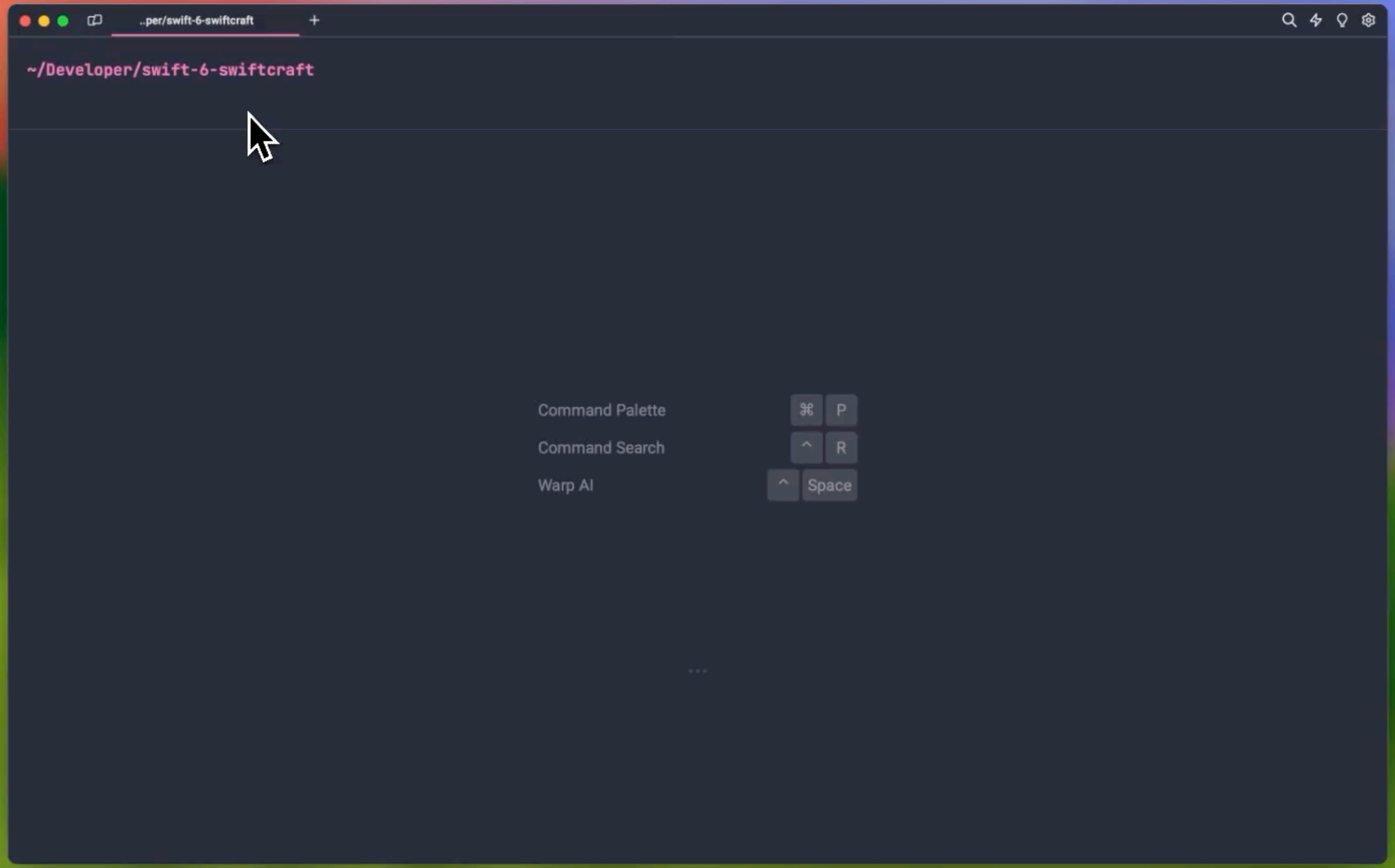
INSTALL ON MACOS

```
#!/bin/sh

# Install `swiftenv`
brew install kylef/formulae/swiftenv
echo 'if which swiftenv > /dev/null; then eval "$(swiftenv init -)"; fi' >> ~/.zshrc

# Install the latest Swift 6 development toolchain
swiftenv install 6.0-DEVELOPMENT-SNAPSHOT-2024-04-30-a

# CD into your Swift package directory
cd your-swift-package

# Set the Swift 6 toolchain as the default for this directory
swiftenv local 6.0-DEVELOPMENT-SNAPSHOT-2024-04-30-a
```

INSTALL ON LINUX

```
#!/bin/sh

# Install swiftly
curl -L https://swift-server.github.io/swiftly/swiftly-install.sh | bash

# Install the latest Swift 6 development toolchain
swiftly install 6.0-DEVELOPMENT-SNAPSHOT-2024-04-30-a

# Set the Swift 6 toolchain as the active toolchain
swiftly use 6.0-DEVELOPMENT-SNAPSHOT-2024-04-30-a
```

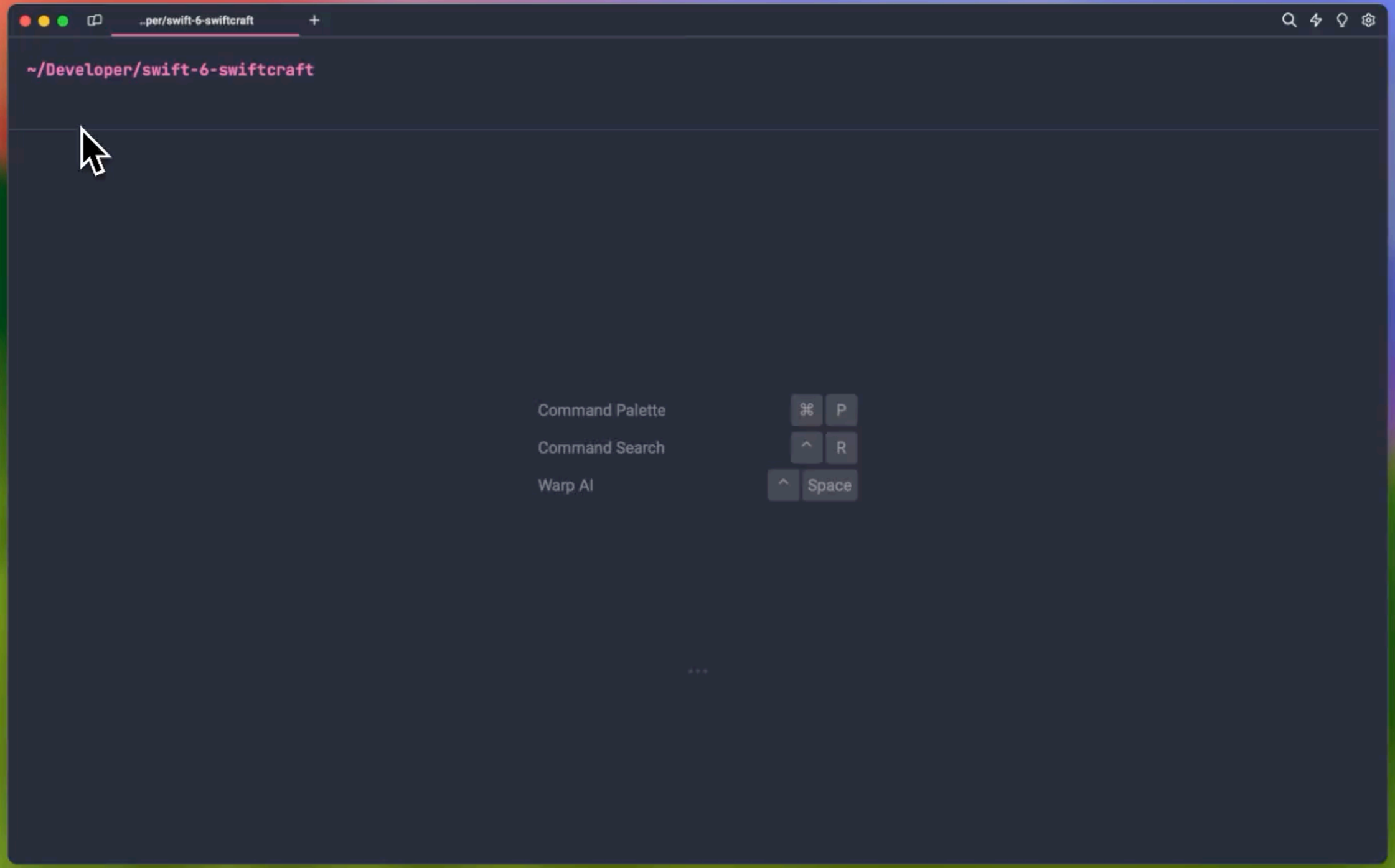
```
class NonIsolated {
    func callee() async {}
}

actor Isolated {
    let nonIsolated = NonIsolated()

    func callee() async {
        await nonIsolated.callee()
    }
}
```


..per/swift-6-swiftcraft
~/Developer/swift-6-swiftcraft
Command Palette
⌘ P
Command Search
^ R
Warp AI
^ Space

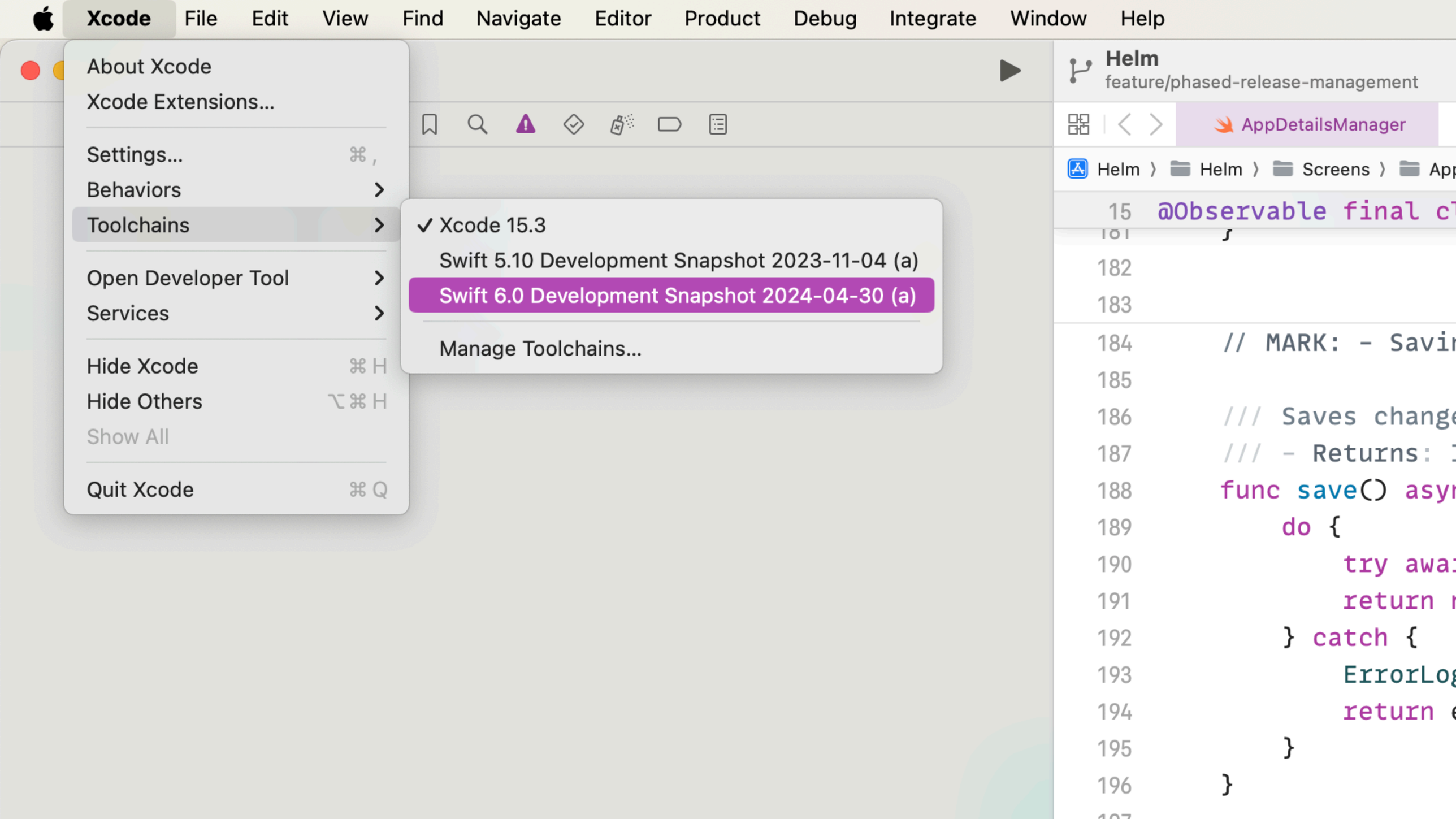
```
// swift-tools-version: 6.0
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    platforms: [.macOS(.v10_15), .iOS(.v13)],
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]
        )
    ],
    targets: [
        .target(name: "Swift6Examples")
    ],
    swiftLanguageVersions: [.version("6")]
)
```

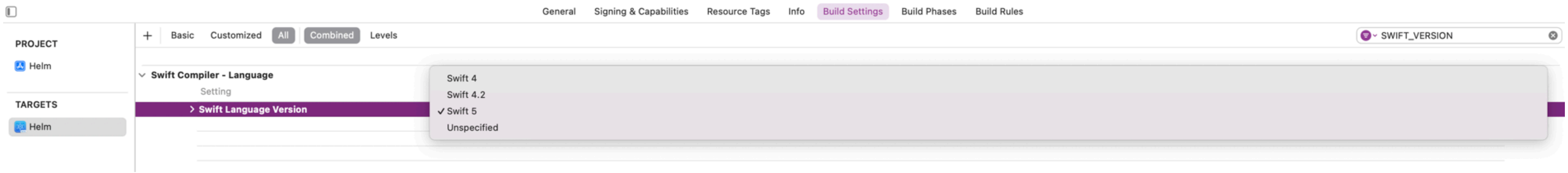
```
// swift-tools-version: 6.0
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    platforms: [.macOS(.v10_15), .iOS(.v13)],
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]
        )
    ],
    targets: [
        .target(name: "Swift6Examples")
    ],
    swiftLanguageVersions: [.version("6")]
)
```

```
swift build -Xswiftc -swift-version -Xswiftc 6
```


..per/swift-6-swiftcraft
~/Developer/swift-6-swiftcraft
Command Palette
⌘ P
Command Search
^ R
Warp AI
^ Space

I HAVE NO IDEA, I DON'T THINK YOU CAN


Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Integrate
Window
Help
About Xcode
Xcode Extensions...
Settings... ⌘ ,
Behaviors
Toolchains
Open Developer Tool
Services
Hide Xcode ⌘ H
Hide Others ⌥ ⌘ H
Show All
Quit Xcode ⌘ Q
✓ Xcode 15.3
Swift 5.10 Development Snapshot 2023-11-04 (a)
Swift 6.0 Development Snapshot 2024-04-30 (a)
Manage Toolchains...
Helm
feature/phased-release-management
AppDetailsManager
Helm 〉 Helm 〉 Screens 〉
15 @Observable final c
182
183
184 // MARK: - Savi
185
186 /// Saves change
187 /// - Returns:
188 func save() asyn
189 do {
190 try awa
191 return
192 } catch {
193 ErrorLo
194 return
195 }
196 }


General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
Helm
TARGETS
Helm
Basic
Customized
All
Combined
Levels
SWIFT_VERSION
Swift Compiler - Language
Setting
Swift Language Version
Swift 4
Swift 4.2
Swift 5
Unspecified

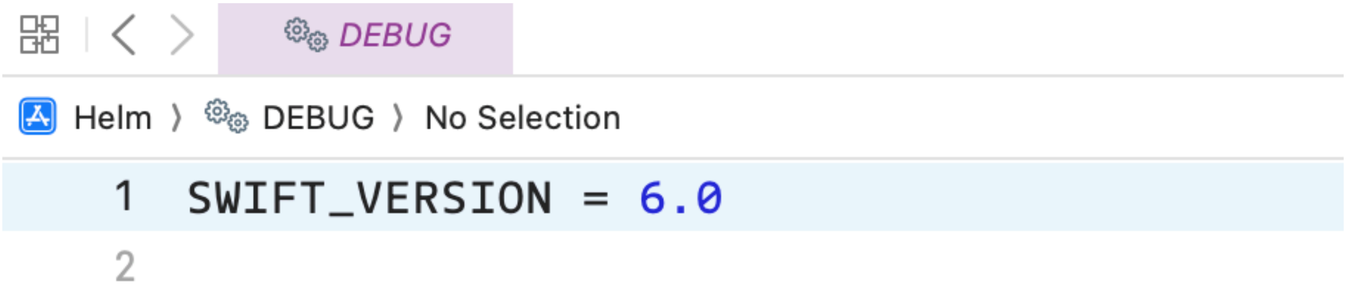
tkremenek Ted Kremenek Dec 2018

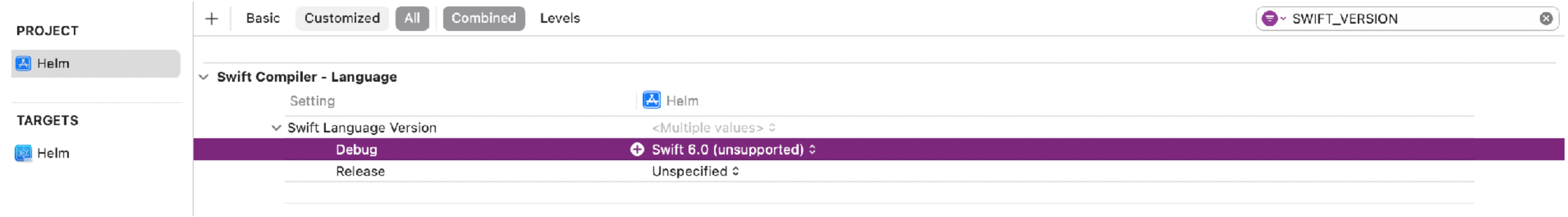
>  Aciid:
>
> Ok, it looks the build system always validates the value of SWIFT_VERSION against the hardcoded list so there is no way make it work with the latest version of Xcode. I'll go ahead and file some radars. We definitely need some solution here :/

Here is a hack that works for trying out the snapshots in Swift 5 mode:

1. Edit the file `/Applications/Xcode.app/Contents/PlugIns/Xcode3Core.ideplugin/Contents/SharedSupport/Developer/Library/Xcode/Plug-ins/XCLanguageSupport.xcplugin/Contents/Resources/Swift.xcspec`
2. Search for `SupportedLanguageVersions` and add 5.0 as a valid value.

11


DEBUG
Helm
DEBUG
No Selection
1 SWIFT_VERSION = 6.0
2


PROJECT
Helm
TARGETS
Helm
Basic
Customized
All
Combined
Levels
SWIFT_VERSION
Swift Compiler - Language
Setting
Helm
Swift Language Version
<Multiple values>
Debug
Swift 6.0 (unsupported)
Release
Unspecified


PROJECT
Helm
TARGETS
Helm
Basic
Customized
All
Combined
Levels
SWIFT_VERSION
Swift Compiler - Language
Setting
Helm
Swift Language Version
Swift 5

UPCOMING FEATURES

HTTPS://GITHUB.COM/APPLE/SWIFT-EVOLUTION/BLOB/MAIN/PROPOSALS/0362-PIECEMEAL-FUTURE-FEATURES.MD

swift-evolution / proposals / 0362-piecemeal-future-features.md

hborla Mark SE-0362 as implemented in Swift 5.8 (#1979) cf8419c · last year History

Preview Code Blame 171 lines (113 loc) · 18.7 KB Raw

Piecemeal adoption of upcoming language improvements

- Proposal: SE-0362
- Authors: Doug Gregor
- Review Manager: Holly Borla
- Status: **Implemented (Swift 5.8)**
- Implementation: apple/swift#59055, apple/swift-package-manager#5632
- Review: (pitch) (review) (acceptance)

Introduction

Swift 6 is accumulating a number of improvements to the language that have enough source-compatibility impact that they could not be enabled by default in prior language modes (Swift 4.x and Swift 5.x). These improvements are already implemented in the Swift compiler behind the Swift 6 language mode, but they are inaccessible to users, and will remain so until Swift 6 becomes available as a language mode. There are several reasons why we should consider making these improvements available sooner:

- Developers would like to get the benefits from these improvements soon, rather than wait until Swift 6 is available.
- Making these changes available to developers prior to Swift 6 provides more experience, allowing us to tune them further for Swift 6 if necessary.
- The sum of all changes made in Swift 6 might make migration onerous for some modules, and adopting these language changes one-by-one while in Swift 4.x/5.x can smooth that transition path.

Implemented

SE-0409 Access-level modifiers on import declarations

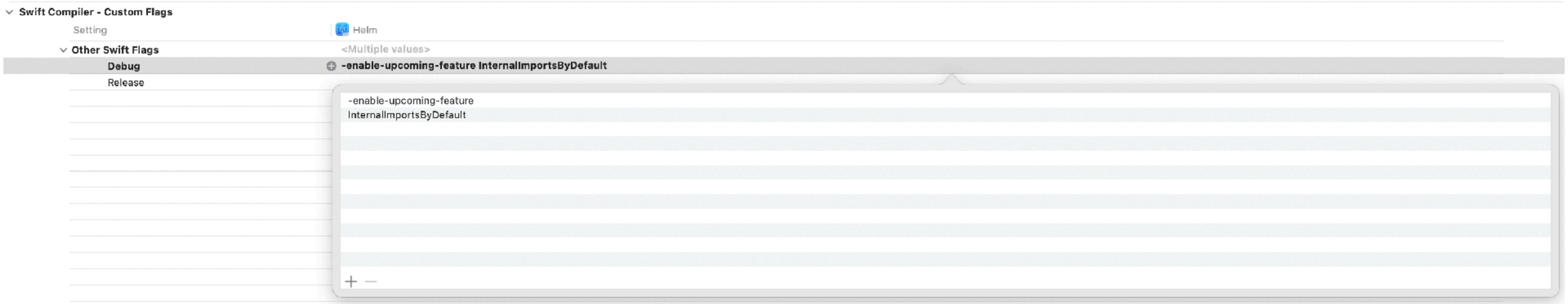
Author: Alexis Laferrière
Review Manager: Frederick Kellison-Linn
Implemented In: Swift 6.0
Upcoming Feature Flag: InternalImportsByDefault

Implemented

SE-0409 Access-level modifiers on import declarations

Author: Alexis Laferrière
Review Manager: Frederick Kellison-Linn
Implemented In: Swift 6.0
Upcoming Feature Flag: InternalImportsByDefault

```swift
// swift-tools-version: 6.0
import PackageDescription

let package = Package(
    name: "UpcomingFeatures",
    products: [
        .library(
            name: "UpcomingFeatures",
            targets: ["UpcomingFeatures"]),
    ],
    dependencies: [
    ],
    targets: [
        .target(
            name: "UpcomingFeatures",
            swiftSettings: [
                .enableUpcomingFeature("InternalImportsByDefault")
            ]
        )
    ]
)
```


Swift Compiler - Custom Flags
Setting
Helm
Other Swift Flags
<Multiple values>
Debug
-enable-upcoming-feature InternalImportsByDefault
Release
-enable-upcoming-feature
InternalImportsByDefault

```
#if hasFeature(InternalImportsByDefault)
fileprivate import MyDependency
#else
import MyDependency
#endif
```

Implicitly Opened Existentials

- Proposal: SE-0352
- Authors: Doug Gregor
- Review Manager: Joe Groff
- Status: **Implemented (Swift 5.7)**
- Upcoming Feature Flag: `ImplicitOpenExistentials` (Implemented in Swift 6.0) (Enabled in Swift 6 language mode)
- Implementation: apple/swift#41996, macOS toolchain
- Decision Notes: Acceptance
- Previous Revision: 1
- Previous Review: First review

Language Mode!

A QUICK TOUR OF THE SWIFT 6

BREAKING CHANGES

The Swift 6 migration

Upcoming features 120 are critical for migrating to the Swift 6 language mode: one can enable individual upcoming features one at a time to incrementally move a code base toward Swift 6, before adopting the full language mode. There are a number of previously-accepted upcoming features that will be enabled by default in Swift 6:

- SE-0274: Concise magic file names 323 (`ConciseMagicFile`)
- SE-0286: Forward-scan matching for trailing closures 163 (`ForwardTrailingClosures`)
- SE-0337: Incremental migration to concurrency checking 75 (`StrictConcurrency`)
- SE-0354: Regex Literals 100 (`BareSlashRegexLiterals`)
- SE-0383: Deprecate @UIApplicationMain and @NSApplicationMain 212 (`DeprecateApplicationMain`)
- SE-0384: Importing Forward Declared Objective-C Interfaces and Protocols 93 (`ImportObjcForwardDeclarations`)
- SE-0401: Remove Actor Isolation Inference caused by Property Wrappers 80 (`DisableOutwardActorInference`)
- SE-0409: Access-level modifiers on import declarations 185 (`InternalImportsByDefault`)

The Language Steering Group has decided that one previously-accepted upcoming feature, `ExistentialAny`, will **not** be enabled by default in Swift 6. SE-0335: Introduce existential any 95 introduces the `any` keyword to identify existential types. The proposal also specifies that "bare" protocol names will no longer be permitted as types---they must either use `any` or `some`, as appropriate---under the upcoming feature flag `ExistentialAny`. Given the concerns about migration to consistent use of existential `any` in the language, and the expectation that additional language improvements will be coming that might affect the end result of that migration, the Language Steering Group is deferring the source-incompatible changes in SE-0335 to a future language revision.

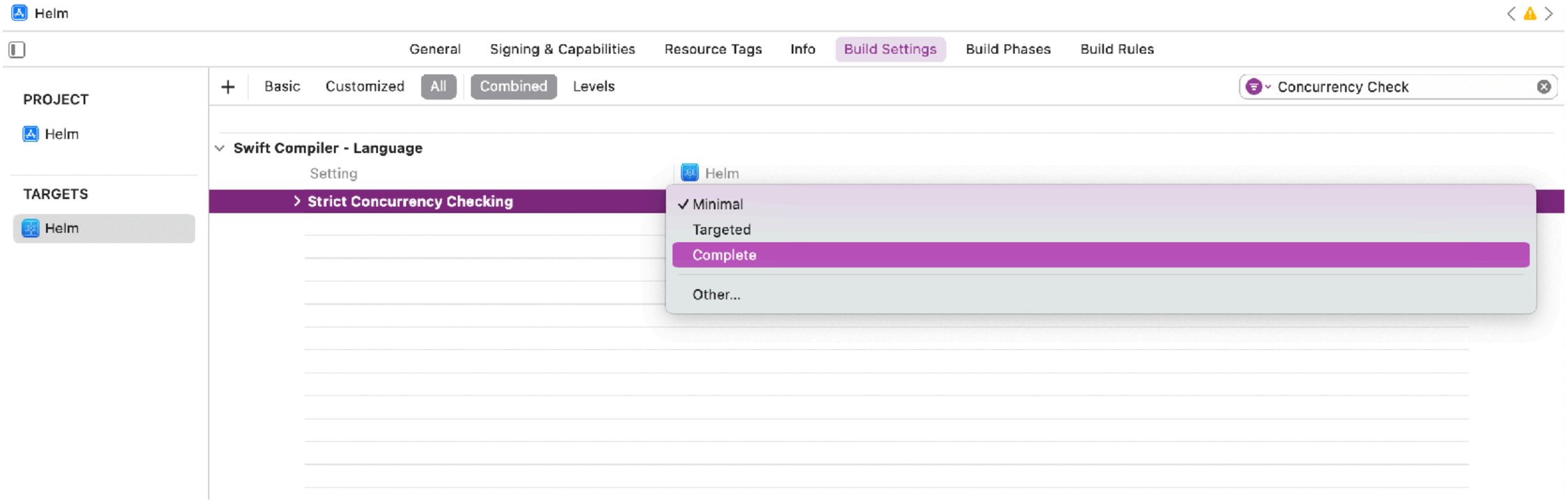
Additional evolution proposals may include source-incompatible changes for Swift 6, and the Language Steering Group will evaluate such proposals on a case-by-case basis for high value to the language relative to the cost of source incompatibilities. For example, sufficiently high value changes might include minor type inference changes that are necessary for completing a major feature, such as typed throws, or semantic changes in pursuit of data race safety.

HTTPS://FORUMS.SWIFT.ORG/T/PROGRESS-TOWARD-THE-SWIFT-6-LANGUAGE-MODE/68315

SE-0337

CONCURRENCY CHECKING

```
// swift-tools-version: 5.10
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    platforms: [.macOS(.v14)],
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    targets: [
        .target(
            name: "Swift6Examples",
            dependencies: ["DependencyExample"],
            swiftSettings: [
                .enableExperimentalFeature("StrictConcurrency")
          ]
        )
    ]
)
```


Helm
General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
Helm
TARGETS
Helm
Basic
Customized
All
Combined
Levels
Concurrency Check
Swift Compiler - Language
Setting
Helm
Strict Concurrency Checking
Minimal
Targeted
Complete
Other...

```swift
import SwiftUI

struct PrivateKeyStepView: View {

    var body: some View {
        VStack(alignment: .leading, spacing: 8) {
            Text("Select a file!")
        }
        .onTapGesture { selectFile() }
        .padding()
    }

    func selectFile() {
        let panel = NSOpenPanel()                                   ⚠ Call to main actor-isolated initializer 'init()' in a synchronous nonisolated context; this is an error in Swift 6
        panel.canChooseFiles = true                                 ⚠ Main actor-isolated property 'canChooseFiles' can not be mutated from a non-isolated context; this is an error in Swift 6
        panel.canChooseDirectories = false                          ⚠ Main actor-isolated property 'canChooseDirectories' can not be mutated from a non-isolated context; this is an error in Swift 6
        panel.allowsMultipleSelection = false                       ⚠ Main actor-isolated property 'allowsMultipleSelection' can not be mutated from a non-isolated context; this is an error in Swift 6

        // Set the initial directory to the Downloads folder
        if let downloadsPath = FileManager.default.urls(for: .downloadsDirectory, in: .userDomainMask).first {
            panel.directoryURL = downloadsPath                      ⚠ Main actor-isolated property 'directoryURL' can not be mutated from a non-isolated context; this is an error in Swift 6
        }

        // Show the open panel and check if the user selected a file
        if panel.runModal() == .OK, let selectedFile = panel.urls.first {   2 ⚠ Call to main actor-isolated instance method 'runModal()' in a synchronous nonisolated context;...
            print("Selected file: \(selectedFile.path)")
        }
    }
}
```

```
import AppStoreConnect_Swift_SDK
```

Add '@preconcurrency' to suppress 'Sendable'-related warnings from module 'AppStoreConnect_Swift_SDK'

I THINK MANY PEOPLE BELIEVE YOU TURN ON COMPLETE CHECKING TO GET PREPARED FOR SWIFT 6. AND, YES, THAT IS TRUE! BUT, I DO NOT THINK THIS IS THE MOST COMPELLING REASON TODAY.

YOU TURN ON COMPLETE CHECKING TO LEARN HOW SWIFT CONCURRENCY ACTUALLY WORKS.

MATT MASSICOTTE

@MATTIEM@MASTODON.SOCIAL

HTTPS://WWW.MASSICOTTE.ORG/COMPLETE-CHECKING

SE-0401

ACTOR ISOLATION PROPERTY WRAPPERS

```
final class Model: ObservableObject {
    let name: String

    init(name: String = "Hello!") {
        self.name = name
    }
}
```

Non-isolated

```
struct MyView: View {
   @StateObject private var model = Model()

    var body: some View {
        Text("Hello, \(model.name)")
            .onAppear { viewAppeared() }
    }

    func viewAppeared() {
        updateUI()
    }
}
```

```
@MainActor func updateUI() { /* do stuff here */ }
```

```
final class Model: ObservableObject {
    let name: String

    init(name: String = "Hello!") {
        self.name = name
    }
}

struct MyView: View {
   @StateObject private var model = Model()

    var body: some View {
        Text("Hello, \(model.name)")
            .onAppear { viewAppeared() }
    }

    func viewAppeared() {
        updateUI()
    }
}

@MainActor func updateUI() { /* do stuff here */ }
```

Isolated to the Global Main Actor

```swift
final class Model: ObservableObject {
    let name: String

    init(name: String = "Hello!") {
        self.name = name
    }
}

struct MyView: View {
    @StateObject private var model = Model()

    var body: some View {
        Text("Hello, \(model.name)")
            .onAppear { viewAppeared() }
    }

    func viewAppeared() {
        updateUI()
    }
}

@MainActor func updateUI() { /* do stuff here */ }
```

Isolated to Main Actor

```
final class Model: ObservableObject {
    let name: String

    init(name: String = "Hello!") {
        self.name = name
    }
}
```

Isolated to Main Actor

```
struct MyView: View {
   @StateObject private var model = Model()

    var body: some View {
        Text("Hello, \(model.name)")
            .onAppear { viewAppeared() }
    }

    func viewAppeared() {
        updateUI()
    }
}
```

```
@MainActor func updateUI() { /* do stuff here */ }
```

```
// swift-tools-version: 5.10
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    platforms: [.macOS(.v14)],
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    targets: [
        .target(
            name: "Swift6Examples",
            dependencies: ["DependencyExample"],
            swiftSettings: [
                .enableExperimentalFeature("StrictConcurrency"),
                .enableUpcomingFeature(“DisableOutwardActorInference")
            ]
        )
    ]
)
```

```swift
final class Model: ObservableObject {
    let name: String

    init(name: String = "Hello!") {
        self.name = name
    }
}

struct MyView: View {
    // Note that `StateObject` has a MainActor-isolated `wrappedValue`
    @StateObject private var model = Model()

    var body: some View {
        Text("Hello, \(model.name)")
            .onAppear { viewAppeared() }
    }

    // This function is inferred to be `@MainActor`
    func viewAppeared() {
        updateUI()
    }
}

@MainActor func updateUI() { /* do stuff here */ }
```

Call to main actor-isolated global function 'updateUI()' in a synchronous nonisolated context

Calls to global function 'updateUI()' from outside of its actor context are implicitly asynchronous

```swift
// swift-tools-version: 6.0
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    platforms: [.macOS(.v14)],
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    targets: [
        .target(
            name: "Swift6Examples",
            dependencies: ["DependencyExample"],
            swiftSettings: [
                .enableExperimentalFeature("StrictConcurrency"),
                .enableUpcomingFeature("DisableOutwardActorInference"),
                .enableExperimentalFeature("RegionBasedIsolation"),
                .enableUpcomingFeature("IsolatedDefaultValues"),
                .enableUpcomingFeature("GlobalConcurrency"),
                .enableUpcomingFeature("InferSendableFromCaptures")
            ]
        )
    ],
    swiftLanguageVersions: [.version("6")]
)
```

SE-0274

CONCISE MAGIC FILE

```
public extension XCTestCase {
  func assertInstancesAreDeallocated(
    _ instances: [AnyObject],
    line: UInt = #line,
    file: StaticString = #file
  ) {
    let instancesContainer = NSHashTable<AnyObject>.weakObjects()
    instances.forEach { instancesContainer.add($0) }

    addTeardownBlock {
      instancesContainer.allObjects.forEach {
        XCTAssertNil($0, file: file, line: line)
      }
    }
  }
}
```

```swift
public extension XCTestCase {
  func assertInstancesAreDeallocated(
    _ instances: [AnyObject],
    line: UInt = #line,
    file: StaticString = #file
  ) {
    let instancesContainer = NSHashTable<AnyObject>.weakObjects()
    instances.forEach { instancesContainer.add($0) }

    addTeardownBlock {
      instancesContainer.allObjects.forEach {
        XCTAssertNil($0, file: file, line: line)
      }
    }
  }
}
```

```
final class Swift6ExamplesTests: XCTestCase {
    func test_WhenInstanceIsDeallocated_ThenItDoesNotLeakMemory() throws {
53      let holder = ClosureHolder()
54      let caller = ClosureCaller(holder: holder)
55
56      caller.leak()
57
58      assertInstancesAreDeallocated([holder, caller])   2  XCTAssertNil failed: "Swift6ExamplesTests.ClosureCaller"
59  }
60  }
```

```
// swift-tools-version: 5.8
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    targets: [
        .testTarget(
            name: "Swift6ExamplesTests",
            dependencies: ["Swift6Examples"],
            swiftSettings: [
                .enableUpcomingFeature("ConciseMagicFile")
            ]
        )
   ]
)
```

```swift
#if hasFeature(ConciseMagicFile)
// In Swift 6 Language Mode
XCTAssertEqual(#file, "Swift6ExamplesTests/Swift6ExamplesTests.swift")
#else
// In Swift 5 Language Mode
XCTAssertEqual(
    #file,
    "…/swift-6-swiftcraft/Tests/Swift6ExamplesTests/Swift6ExamplesTests.swift"
)
#endif
```

```
#if hasFeature(ConciseMagicFile)
// In Swift 6 Language Mode
XCTAssertEqual(#file, "Swift6ExamplesTests/Swift6ExamplesTests.swift")
#else
// In Swift 5 Language Mode
XCTAssertEqual(
    #file,
    "…/swift-6-swiftcraft/Tests/Swift6ExamplesTests/Swift6ExamplesTests.swift"
)
#endif
```

```swift
public extension XCTestCase {
  func assertInstancesAreDeallocated(
    _ instances: [AnyObject],
    line: UInt = #line,
    file: StaticString = #file
  ) {
    let instancesContainer = NSHashTable<AnyObject>.weakObjects()
    instances.forEach { instancesContainer.add($0) }

    addTeardownBlock {
      instancesContainer.allObjects.forEach {
        XCTAssertNil($0, file: file, line: line)
      }
    }
  }
}
```

⚠ Parameter 'file' with default argument '#file' passed to parameter 'file', whose default argument is '#filePath'

SE-0354

REGEX LITERALS

```
let pattern = #"WIFI:S:(?<ssid>[^;]+);(?:T:(?<security>[^;]*);)?P:(?<password>[^;]+);(?:H:(?<hidden>[^;]*);)?;"#

let regularExpression = try! NSRegularExpression(pattern: pattern)
```

```
// swift-tools-version: 5.7
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    platforms: [.macOS(.v14)],
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    targets: [
        .target(
            name: "Swift6Examples",
            dependencies: ["DependencyExample"],
            swiftSettings: [
                .enableUpcomingFeature("BareSlashRegexLiterals")
          ]
        )
    ]
)
```

```
let regex = /WIFI:S:(?<ssid>[^;]+);(?:T:(?<security>[^;]*);)?P:(?<password>[^;]+);(?:H:(?<hidden>[^;]*);)?;/
```

SE-0286

TRAILING CLOSURES

```
struct TestStructure {
    let firstClosure: ((String) → Void)?
    let secondClosure: ((String, Int) → Void)?

    init(
        firstClosure: ((String) → Void)? = nil,
        secondClosure: ((String, Int) → Void)? = nil
    ) {
        self.firstClosure = firstClosure
        self.secondClosure = secondClosure
    }
}
```

```
struct TestStructure {
    let firstClosure: ((String) → Void)?
    let secondClosure: ((String, Int) → Void)?

    init(
        firstClosure: ((String) → Void)? = nil,
        secondClosure: ((String, Int) → Void)? = nil
    ) {
        self.firstClosure = firstClosure
        self.secondClosure = secondClosure
    }
}
```

```
struct TestStructure {
    let firstClosure: ((String) → Void)?
    let secondClosure: ((String, Int) → Void)?

    init(
        firstClosure: ((String) → Void)? = nil,
        secondClosure: ((String, Int) → Void)? = nil
    ) {
        self.firstClosure = firstClosure
        self.secondClosure = secondClosure
    }
}
```

```
let a = TestStructure { a in
    print(a)
}
```

```
struct TestStructure {
    let firstClosure: ((String) → Void)?
    let secondClosure: ((String, Int) → Void)?

    init(
        firstClosure: ((String) → Void)? = nil,
        secondClosure: ((String, Int) → Void)? = nil
    ) {
        self.firstClosure = firstClosure
        self.secondClosure = secondClosure
    }
}
```

```
let a = TestStructure { a in
    print(a)
}
```

Warning!

```
let b = TestStructure { a, b in
    print(a, b)
}
```

```
// swift-tools-version: 5.10
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    platforms: [.macOS(.v14)],
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    targets: [
        .target(
            name: "Swift6Examples",
            dependencies: ["DependencyExample"],
            swiftSettings: [
                .enableUpcomingFeature(“ForwardTrailingClosures")
          ]
        )
    ]
)
```

```swift
struct TestStructure {
    let firstClosure: ((String) → Void)?
    let secondClosure: ((String, Int) → Void)?

    init(
        firstClosure: ((String) → Void)? = nil,
        secondClosure: ((String, Int) → Void)? = nil
    ) {
        self.firstClosure = firstClosure
        self.secondClosure = secondClosure
    }
}
```

Ok!

```swift
let a = TestStructure { a in
    print(a)
}
```

Error!

```swift
let b = TestStructure { a, b in
    print(a, b)
}
```

Ok!

```swift
let c = TestStructure(secondClosure: { a, b in
    print(a, b)
})
```

SE-0352

IMPLICIT OPEN EXISTENTIALS

```
protocol WithSelfRequirement: Equatable {}
```

```
protocol WithSelfRequirement: Equatable {}

func testOpenExistential(selfRequirementProtocol: WithSelfRequirement) {

}
```

Error!

```
protocol WithSelfRequirement: Equatable {}

func testOpenExistential(selfRequirementProtocol: any WithSelfRequirement) {

}
```

```
protocol WithSelfRequirement: Equatable {}


func testOpenExistential(selfRequirementProtocol: any WithSelfRequirement) {

}
```

```swift
protocol WithSelfRequirement: Equatable {}

func requiresOpeningOfExistential<T: WithSelfRequirement>(openExistential: T) {
}

func testOpenExistential(selfRequirementProtocol: any WithSelfRequirement) {

}
```

```swift
protocol WithSelfRequirement: Equatable {}

func requiresOpeningOfExistential<T: WithSelfRequirement>(openExistential: T) {
}

func testOpenExistential(selfRequirementProtocol: any WithSelfRequirement) {
    requiresOpeningOfExistential(openExistential: selfRequirementProtocol)
}
```

bjhomer 10d

Hm… are you sure the upcoming feature flag is even needed? I just dropped the example code from the proposal into an empty swift file, and it compiles and runs without error with Swift 5.10. My test case:

```
protocol P {
    associatedtype A
    func getA() -> A
}

struct S: P {
    func getA() -> Int {
        42
    }
}

func takeP<T: P>(_ value: T) {
    print(value.getA())
}

func test(p: any P) {
    takeP(p)  // no error here anymore!
}

let s = S()
test(p: s)
```

```swift
func printType<ExpectedType>(_ type: ExpectedType.Type) {
    print(type)
}

let type: any Any.Type = Int.self
printType(type) Error!
```

```
// swift-tools-version: 6.0
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    targets: [
        .target(
            name: "Swift6Examples",
            swiftSettings: [
                .enableUpcomingFeature(“ImplicitOpenExistentials”)
            ]
        )
   ]
)
```

SE-0383

DEPRECATE APPLICATIONMAIN

```
import AppKit
@NSApplicationMain
final class MyApplication: NSResponder, NSApplicationDelegate {
    // ...
}

import UIKit
@UIApplicationMain
final class MyApplication: UIResponder, UIApplicationDelegate {
    // ...
}
```

```
import AppKit
@NSApplicationMain
final class MyApplication: NSResponder, NSApplicationDelegate {
    // ...
}

import UIKit
@UIApplicationMain
final class MyApplication: UIResponder, UIApplicationDelegate {
    // ...
}
```

```swift
import AppKit
@main
final class MyApplication: NSResponder, NSApplicationDelegate {
    // ...
}

import UIKit
@main
final class MyApplication: UIResponder, UIApplicationDelegate {
    // ...
}
```

```
// swift-tools-version: 5.10
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    targets: [
        .target(
            name: "Swift6Examples",
            swiftSettings: [
                .enableUpcomingFeature("DeprecateApplicationMain")
            ]
        )
   ]
)
```

```
import AppKit

@NSApplicationMain
final class MyApplication: NSResponder, NSApplicationDelegate {
    // ...
}
```

'NSApplicationMain' is deprecated; this is an error in Swift 6

SE-0409

INTERNAL IMPORTS

```
import DependencyExample
```

```
private
package
public import DependencyExample
internal
fileprivate
```

```
private import DependencyExample
import SwiftUI

public func notAllowed() → DependencyExample  {
    DependencyExample()
}
```

Function cannot be declared public because its result uses a private type

```
// swift-tools-version: 5.9
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    dependencies: [
    ],
    targets: [
        .target(
            name: "Swift6Examples",
            dependencies: ["DependencyExample"],
            swiftSettings: [
                .enableExperimentalFeature("AccessLevelOnImport"),
            ]
        ),
        .target(name: "DependencyExample")
    ]
)
```

Swift 5 language mode

```
public import DependencyExample == import DependencyExample
```

Swift 5 language mode

```
public import DependencyExample == import DependencyExample
```

Swift 6 language mode

```
internal import DependencyExample == import DependencyExample
```

```
// swift-tools-version: 6.0
import PackageDescription

let package = Package(
    name: "Swift6Examples",
    products: [
        .library(
            name: "Swift6Examples",
            targets: ["Swift6Examples"]),
    ],
    dependencies: [
    ],
    targets: [
        .target(
            name: "Swift6Examples",
            dependencies: ["DependencyExample"],
            swiftSettings: [
                .enableExperimentalFeature("AccessLevelOnImport"),
                .enableUpcomingFeature("InternalImportsByDefault")
            ]
        ),
        .target(name: "DependencyExample")
    ]
)
```

```
import DependencyExample
import SwiftUI

public func notAllowed() → DependencyExample  {
    DependencyExample()
}
```

Function cannot be declared public because its result uses an internal type

```swift
#!/usr/bin/swift

private import Foundation

let fileManager = FileManager.default
let currentDirectory = fileManager.currentDirectoryPath
let swiftFiles = fileManager.enumerator(atPath: currentDirectory)?
    .compactMap { $0 as? String }
    .filter { $0.hasSuffix(".swift") }

for file in swiftFiles ?? [] {
    let filePath = "\(currentDirectory)/\(file)"
    guard let content = try? String(contentsOfFile: filePath) else {
        continue
    }

    let updatedContent = content
        .replacingOccurrences(of: #"import (\w+)"#, with: "public import $1",
options: .regularExpression)

    try? updatedContent.write(toFile: filePath, atomically: true, encoding: .utf8)
}
```

HTTPS://WWW.POLPIELLA.DEV/SWIFT-6-IMPORT-ACCESS-LEVEL#ADOPTING-THESE-CHANGES

@_implementationOnly?

SE-0384

OBJ-C FORWARD DECLARATIONS

Below is a list of all the current and upcoming proposal reviews.

13 proposals
Implemented

Implemented

SE-0426 BitwiseCopyable

Authors: Kavon Farvardin, Guillaume Lessard, Nate Chandler, Tim Kientzle
Review Manager: Tony Allevato
Implemented In: Swift 6.0

Implemented

SE-0422 Expression macro as caller-side default argument

Author: Apollo Zhu
Review Manager: Doug Gregor
Implemented In: Swift 6.0
Implementation: swift#70602

ALL BREAKING FEATURES

SE-0412

FEATURE FLAG	SWIFT VERSION
GLOBALCONCURRENCY	5.10
EXPERIMENTAL FEATURE	**SWIFT**
GLOBALCONCURRENCY	5.10

SE-0274

FEATURE FLAG	SWIFT
CONCISEMAGICFILE	5.8

SE-0418

FEATURE FLAG	SWIFT
INFERSENDABLEFROMCAPTURES	6.0

SE-0411

FEATURE FLAG	SWIFT
ISOLATEDDEFAULTVALUES	5.10

SE-0337

FEATURE FLAG	SWIFT
STRICTCONCURRENCYCHECKING	5.6

SE-0383

FEATURE FLAG	SWIFT
DEPRECATEAPPLICATIONMAIN	5.10

SE-0409

FEATURE FLAG	SWIFT VERSION
INTERNALIMPORTSBYDEFAULT	6.0
EXPERIMENTAL FEATURE	**SWIFT**
ACCESSLEVELONIMPORT	5.9

SE-0286

FEATURE FLAG	SWIFT
FORWARDTRAILINGCLOSURES	5.8

SE-0401

FEATURE FLAG	SWIFT
DISABLEOUTWARDACTORINFERENCE	5.9

SE-0352

FEATURE FLAG	SWIFT
IMPLICITOPENEXISTENTIALS	6.0

SE-0384

FEATURE FLAG	SWIFT
IMPORTOBJCFORWARDDECLARATIONS	5.9

SE-0354

FEATURE FLAG	SWIFT
BARESLASHREGEXLITERALS	5.8

THANK YOU!

ANY QUESTIONS?

Download Helm

Find me online!