What's New in Swift

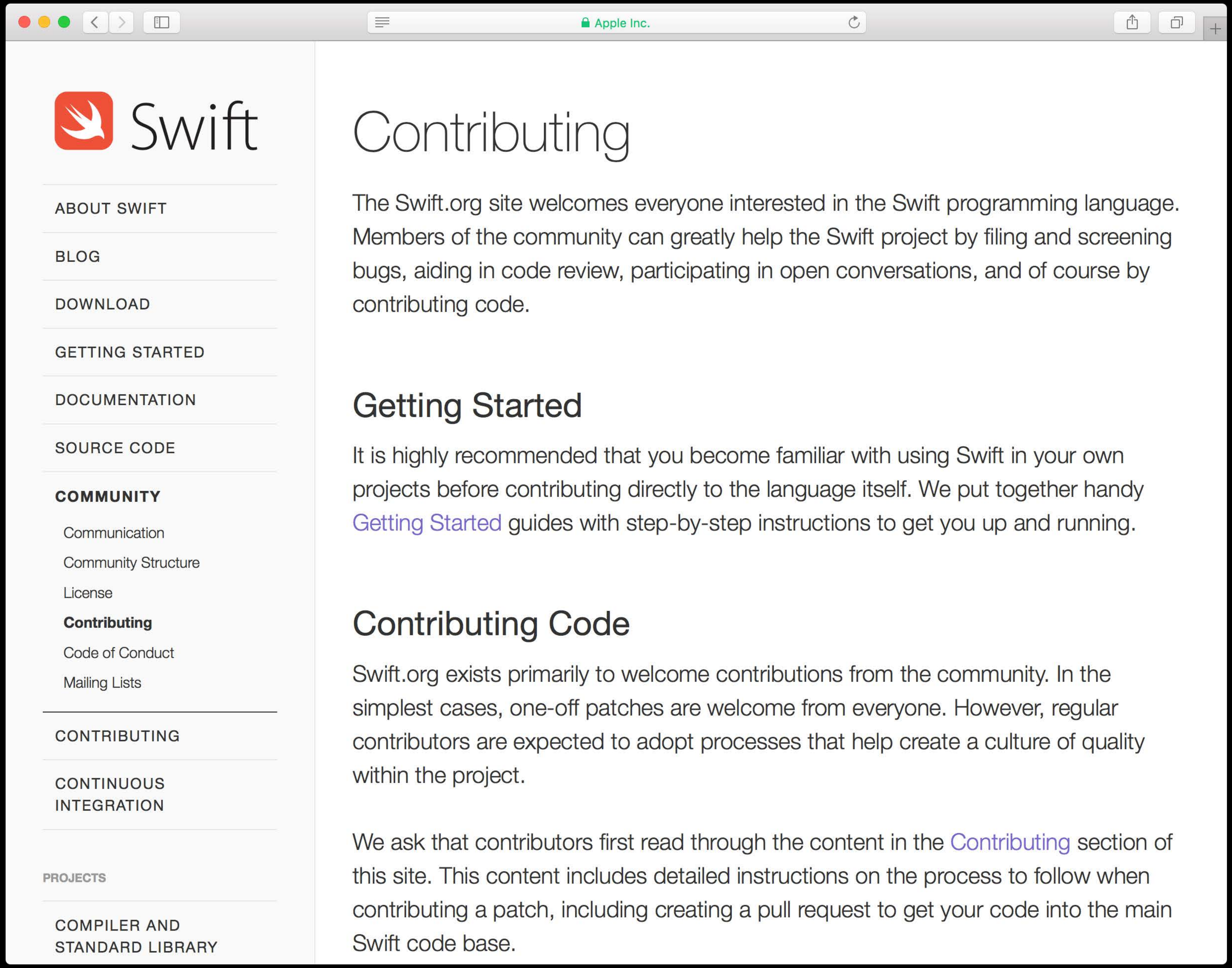
Session 402

Ted Kremenek
Chris Lattner
Ewa Matejska

Goals for Swift 3

Goals for Swift 3

Develop an open community



Contributing

The Swift.org site welcomes everyone interested in the Swift programming language. Members of the community can greatly help the Swift project by filing and screening bugs, aiding in code review, participating in open conversations, and of course by contributing code.

Getting Started

It is highly recommended that you become familiar with using Swift in your own projects before contributing directly to the language itself. We put together handy Getting Started guides with step-by-step instructions to get you up and running.

Contributing Code

Swift.org exists primarily to welcome contributions from the community. In the simplest cases, one-off patches are welcome from everyone. However, regular contributors are expected to adopt processes that help create a culture of quality within the project.

We ask that contributors first read through the content in the Contributing section of this site. This content includes detailed instructions on the process to follow when contributing a patch, including creating a pull request to get your code into the main Swift code base.

Goals for Swift 3

Develop an open community

Portability to new platforms

Goals for Swift 3

Develop an open community

Portability to new platforms

Get the fundamentals right

Goals for Swift 3

- Develop an open community
- Portability to new platforms
- Get the fundamentals right
- Optimize for awesomeness

MacBook

Swift Adoption at Apple

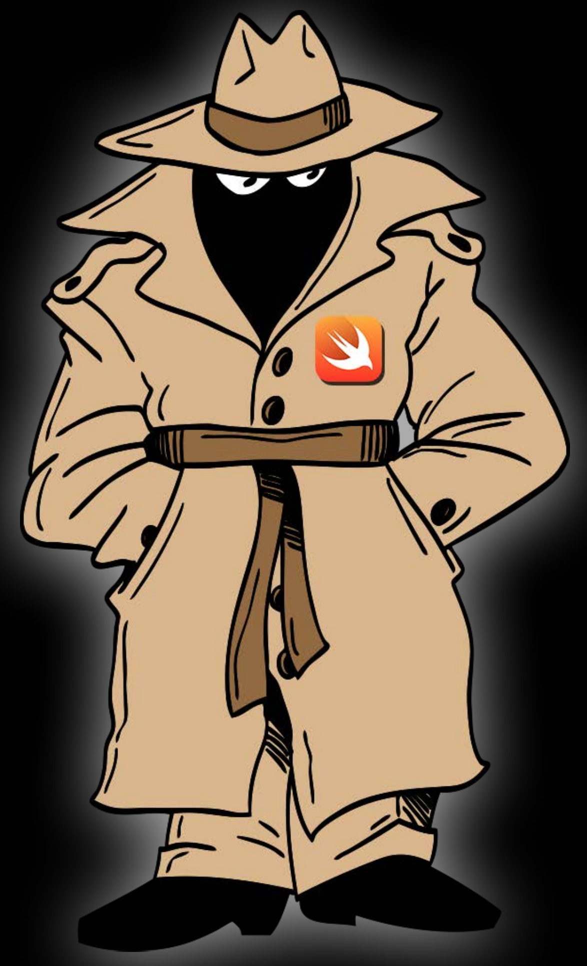
Swift Adoption at Apple

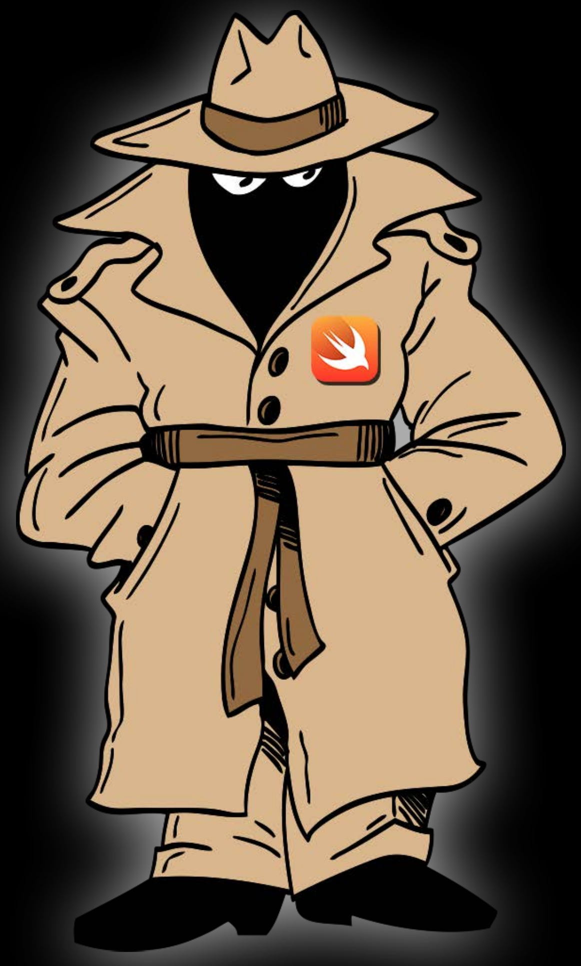
Music

Console

Agents and Daemons

WARNIN
AY 7:36





Dock

Dock

Dock Bar at Bottom

Mission Control

LaunchPad

Command-Tab Application Switcher

Stacks

Accelerated Two Up

Dashboard

Spaces

Some of Notification System

What Changed from El Capitan to Sierra?

Most of Mission Control completely rewritten in Swift

Accessibility engine completely rewritten in Swift

Project Evolution

Lines of code

Project Evolution

Lines of code

Dock is ~200,000 lines of code

Project Evolution

Lines of code

Dock is ~200,000 lines of code

2.5x more Swift code

Project Evolution

Lines of code

Dock is ~200,000 lines of code

2.5x more Swift code

15% less code to rewrite the same functionality in Swift

Project Evolution

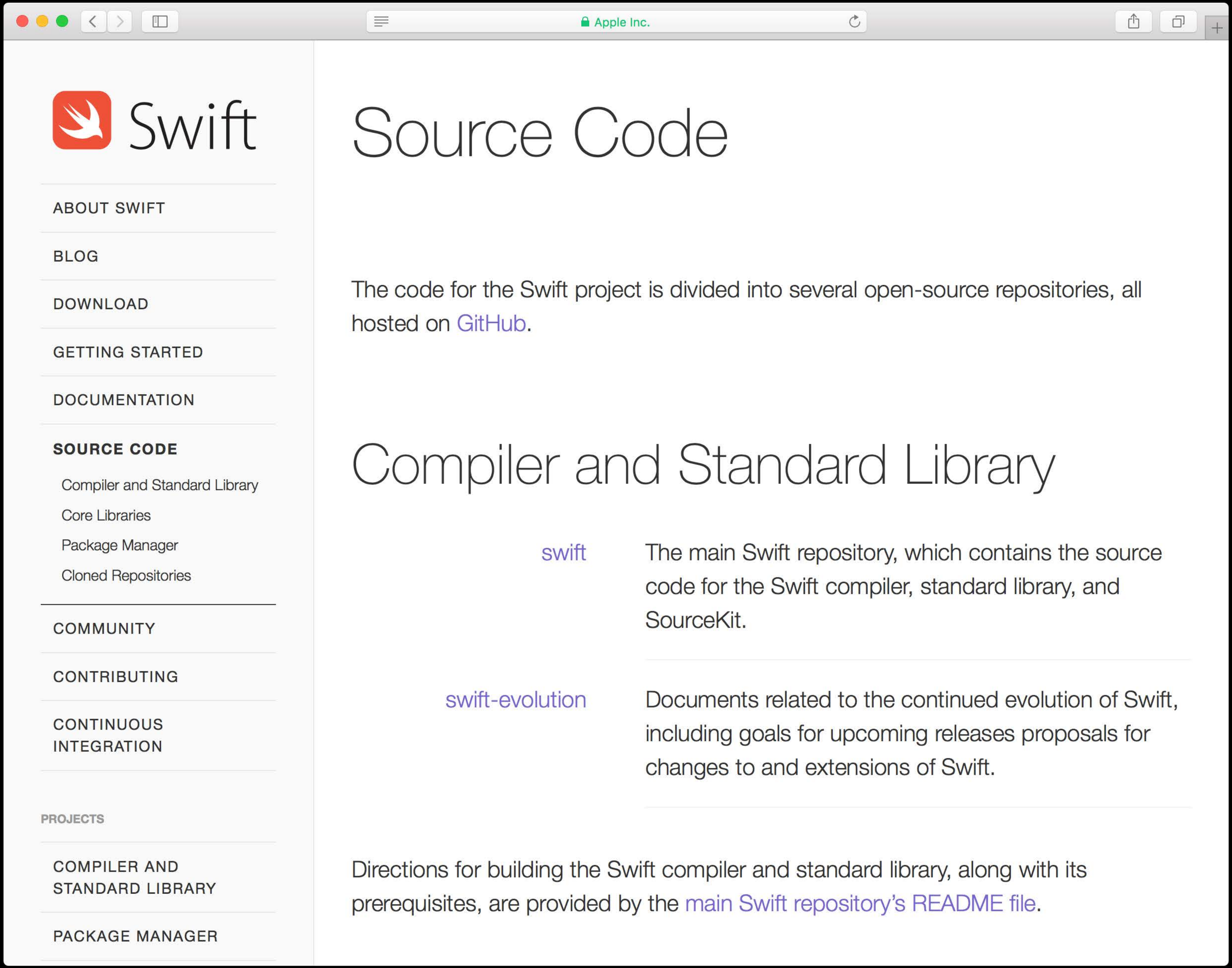
Lines of code

Dock is ~200,000 lines of code

2.5x more Swift code

15% less code to rewrite the same functionality in Swift

New features were added at the same time swift.org



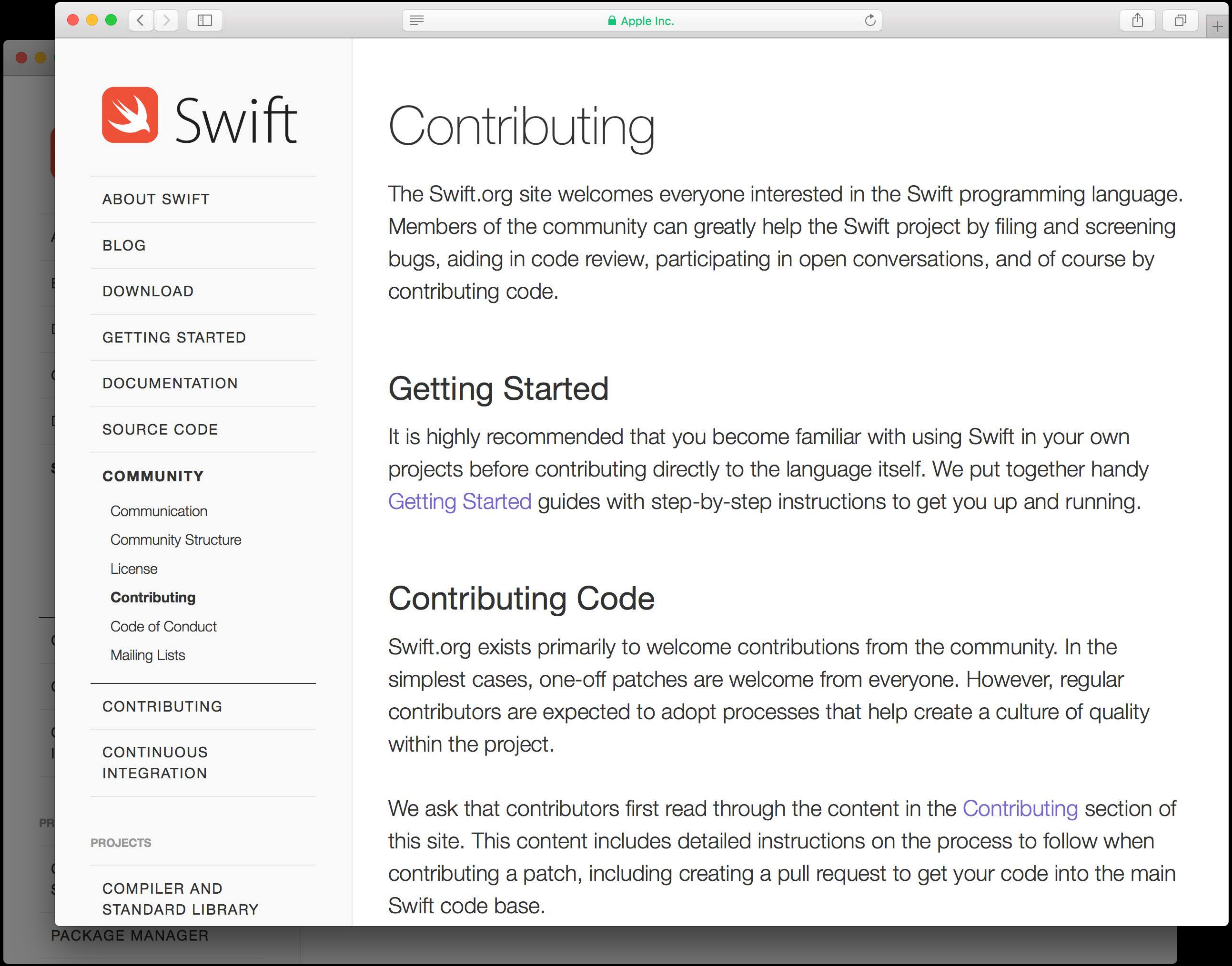
Source Code

The code for the Swift project is divided into several open-source repositories, all hosted on GitHub.

Compiler and Standard Library

swift	The main Swift repository, which contains the source code for the Swift compiler, standard library, and SourceKit.
swift-evolution	Documents related to the continued evolution of Swift, including goals for upcoming releases proposals for changes to and extensions of Swift.

Directions for building the Swift compiler and standard library, along with its prerequisites, are provided by the main Swift repository's README file.



Contributing

The Swift.org site welcomes everyone interested in the Swift programming language. Members of the community can greatly help the Swift project by filing and screening bugs, aiding in code review, participating in open conversations, and of course by contributing code.

Getting Started

It is highly recommended that you become familiar with using Swift in your own projects before contributing directly to the language itself. We put together handy Getting Started guides with step-by-step instructions to get you up and running.

Contributing Code

Swift.org exists primarily to welcome contributions from the community. In the simplest cases, one-off patches are welcome from everyone. However, regular contributors are expected to adopt processes that help create a culture of quality within the project.

We ask that contributors first read through the content in the Contributing section of this site. This content includes detailed instructions on the process to follow when contributing a patch, including creating a pull request to get your code into the main Swift code base.

Swift Open Source

Open evolution process

Non-Apple contributors with direct commit access

Code of conduct

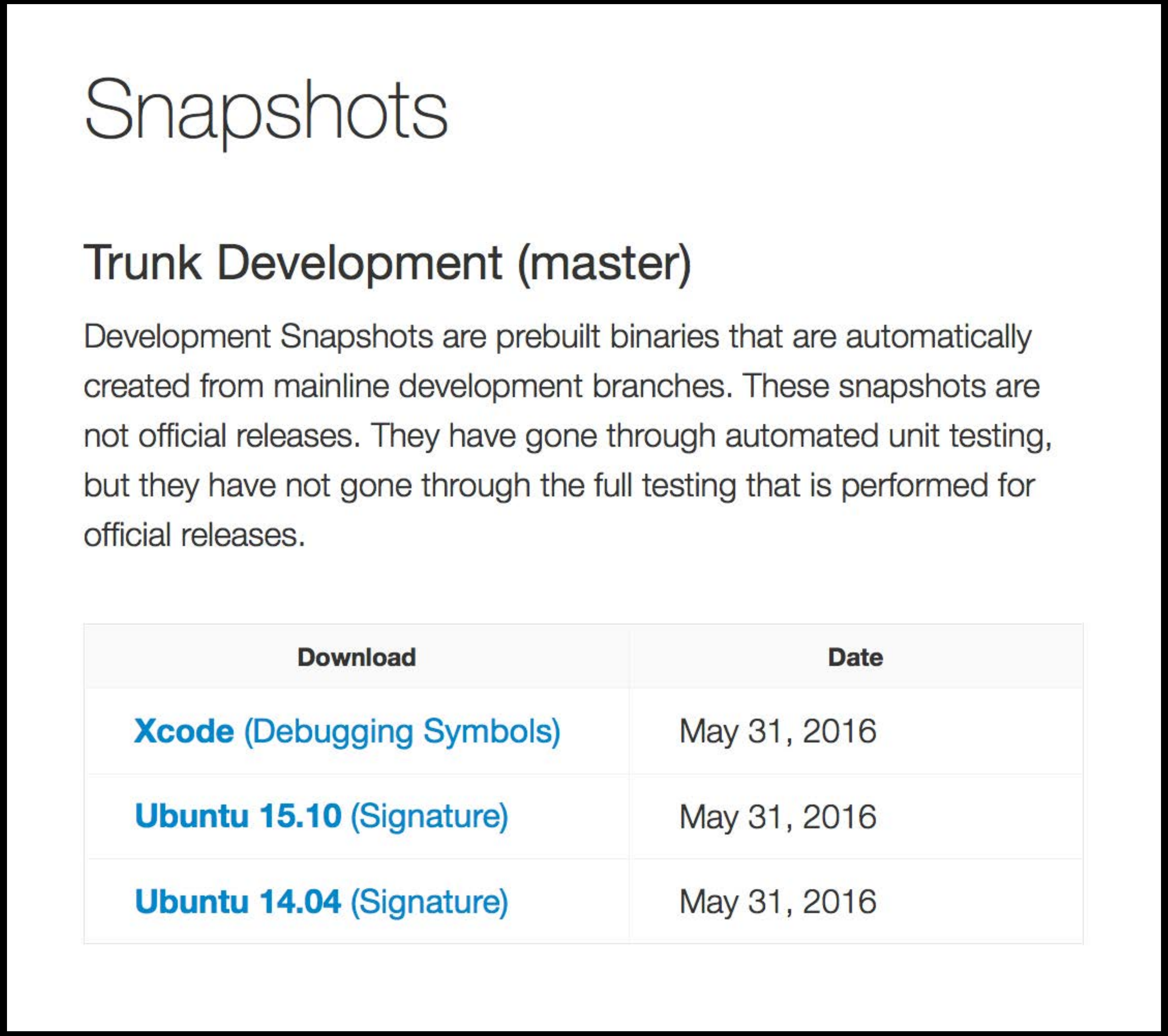
Apache 2 with Runtime Library Exception

Downloadable Toolchains

Download toolchains as Swift develops!

- Xcode (Apple platforms) and Linux
- Built by continuous integration system

Playground support in Xcode 8 (coming soon)



Snapshots

Trunk Development (master)

Development Snapshots are prebuilt binaries that are automatically created from mainline development branches. 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
Xcode (Debugging Symbols)	May 31, 2016
Ubuntu 15.10 (Signature)	May 31, 2016
Ubuntu 14.04 (Signature)	May 31, 2016

GitHub

Swift.org Projects on GitHub

swift

swift-llbuild

swift-lldb

swift-corelibs-xctest

swift-llvm

Swift.org Projects on **GitHub**

swift-evolution

swift-package-manager

swift-corelibs-foundation

swift-corelibs-libdispatch

swift-clang

swift

swift-llbuild

swift-lldb

swift-corelibs-xctest

swift-llvm

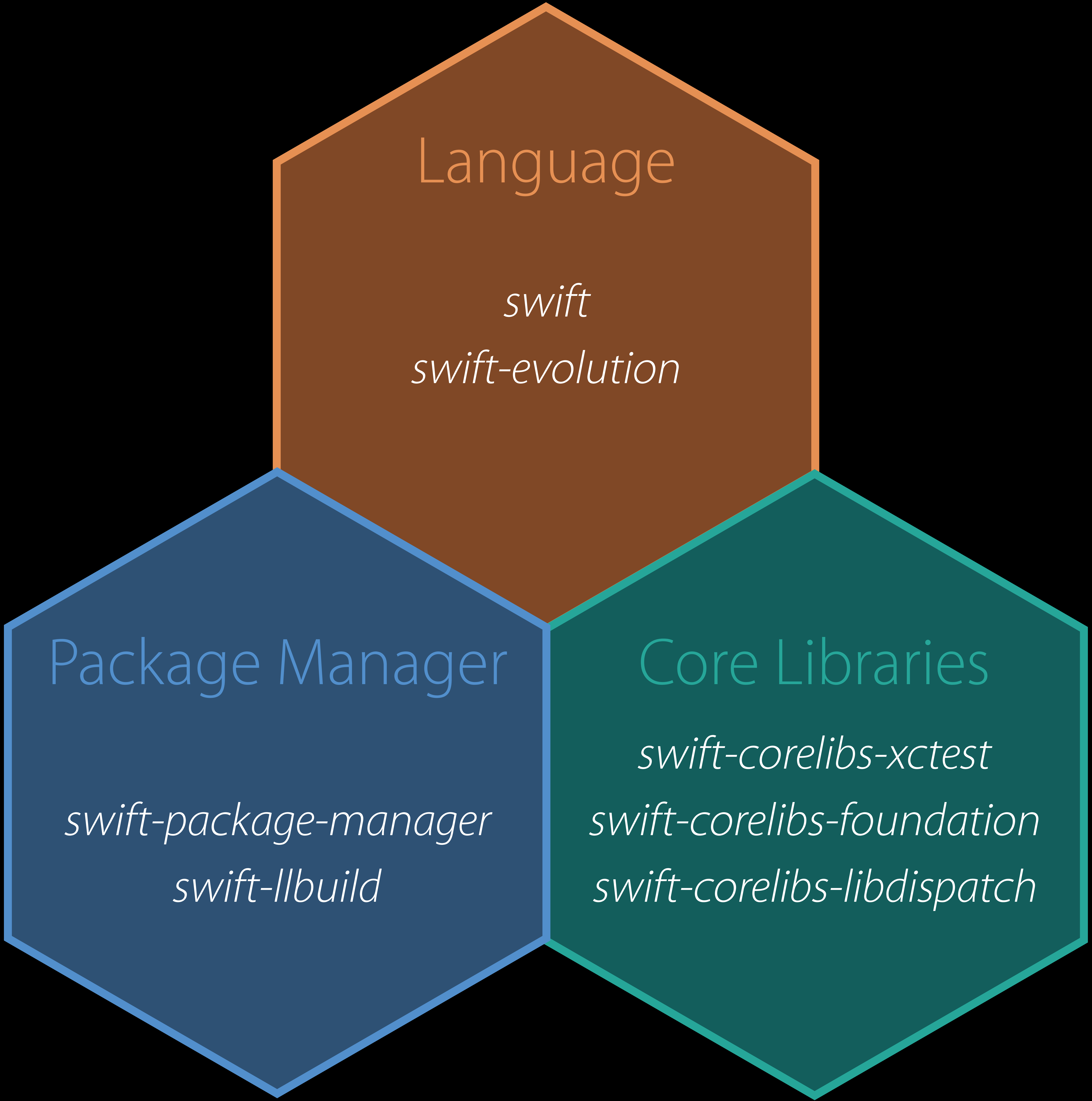
swift-evolution

swift-package-manager

swift-corelibs-foundation

swift-corelibs-libdispatch

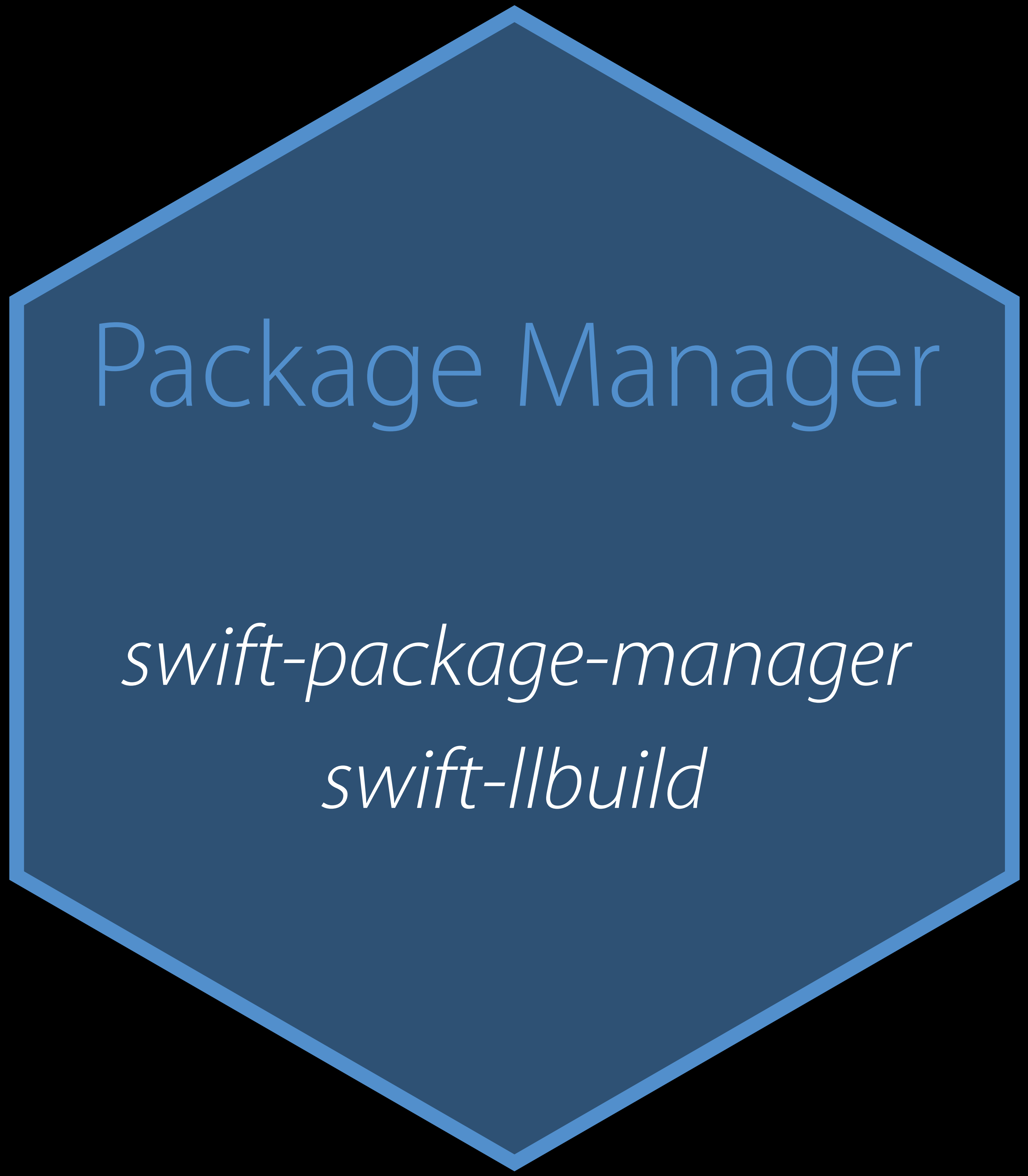
swift-clang


Language
swift
swift-evolution
Package Manager
swift-package-manager
swift-llbuild
Core Libraries
swift-corelibs-xctest
swift-corelibs-foundation
swift-corelibs-libdispatch

Swift Package Manager

Package Manager
swift-package-manager
swift-llbuild

Swift Package Manager



Early and actively in development

Swift Package Manager

Early and actively in development

Cross-platform packages

Swift Package Manager

Early and actively in development

Cross-platform packages

Designed for frictionless development

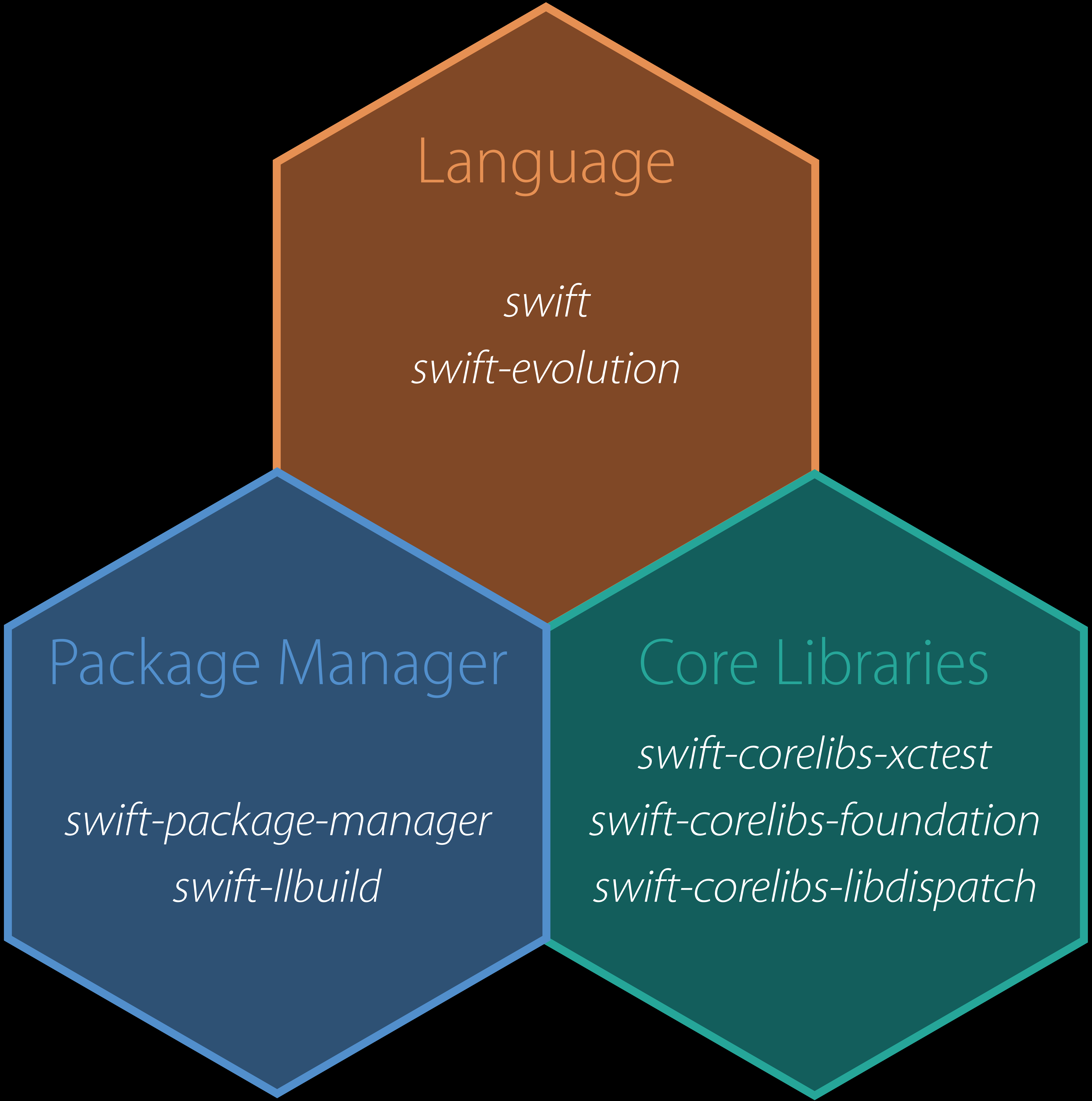
Swift Package Manager

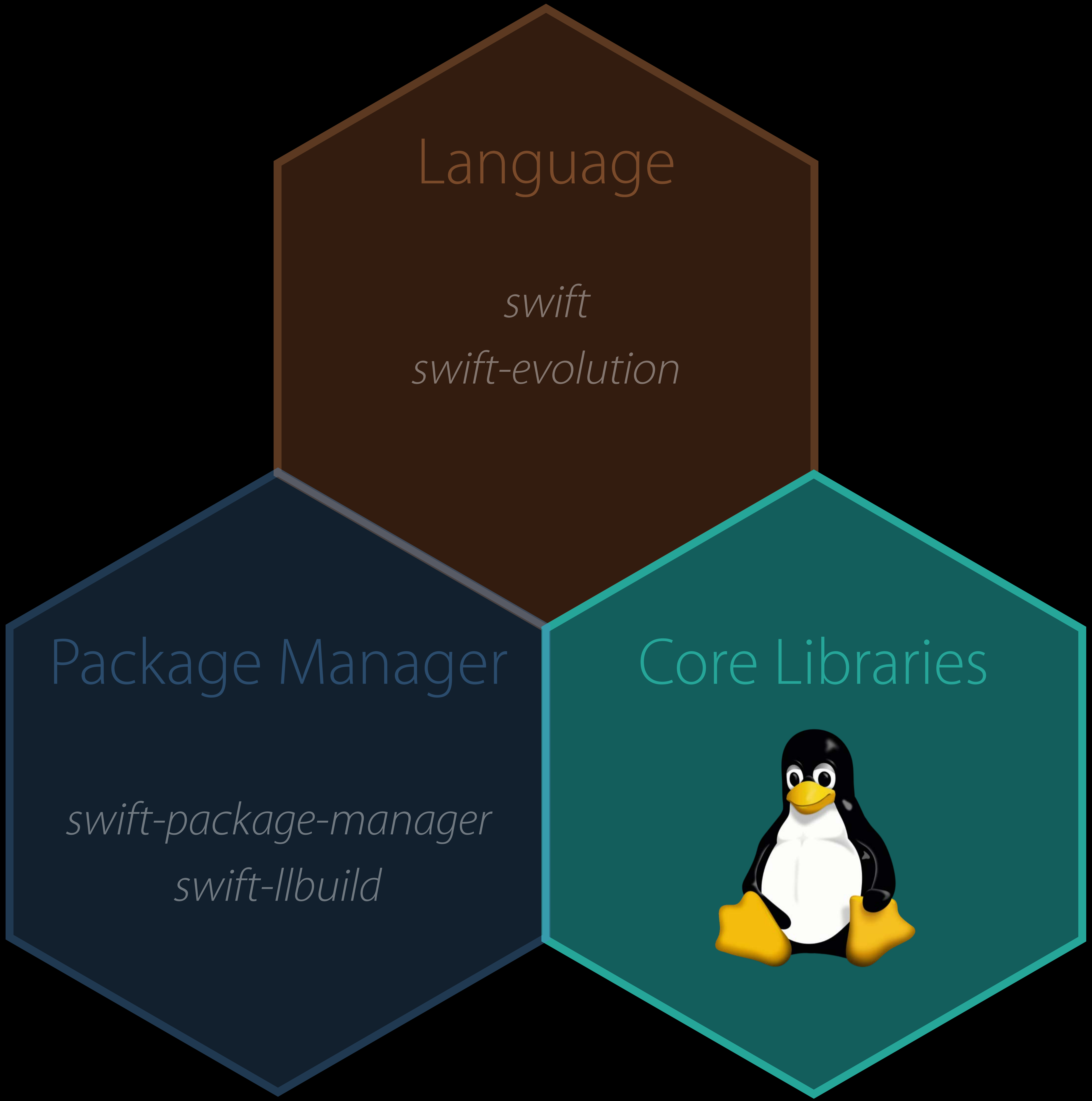
Early and actively in development

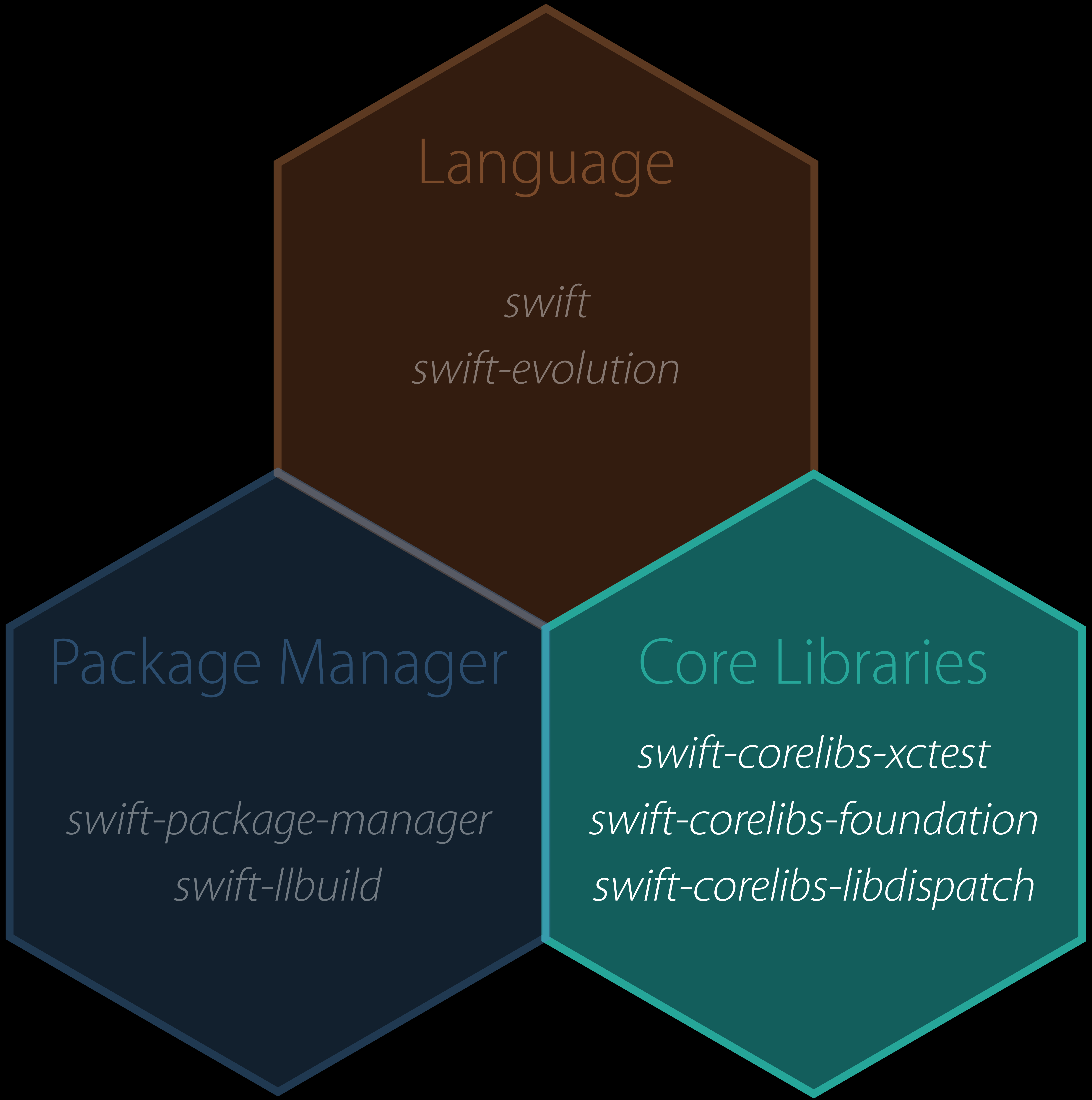
Cross-platform packages

Designed for frictionless development

Great Xcode integration in the future


Language
swift
swift-evolution
Package Manager
swift-package-manager
swift-llbuild
Core Libraries
swift-corelibs-xctest
swift-corelibs-foundation
swift-corelibs-libdispatch


Language
swift
swift-evolution
Package Manager
swift-package-manager
swift-llbuild
Core Libraries


Language
swift
swift-evolution
Package Manager
swift-package-manager
swift-llbuild
Core Libraries
swift-corelibs-xctest
swift-corelibs-foundation
swift-corelibs-libdispatch

Foundation on Linux

Foundation on Linux

URLRequest

UUID

URLQueryItem

PersonNameComponents

AffineTransform

Measurement

CharacterSet

Data

URLComponents

Notification

URL

Date

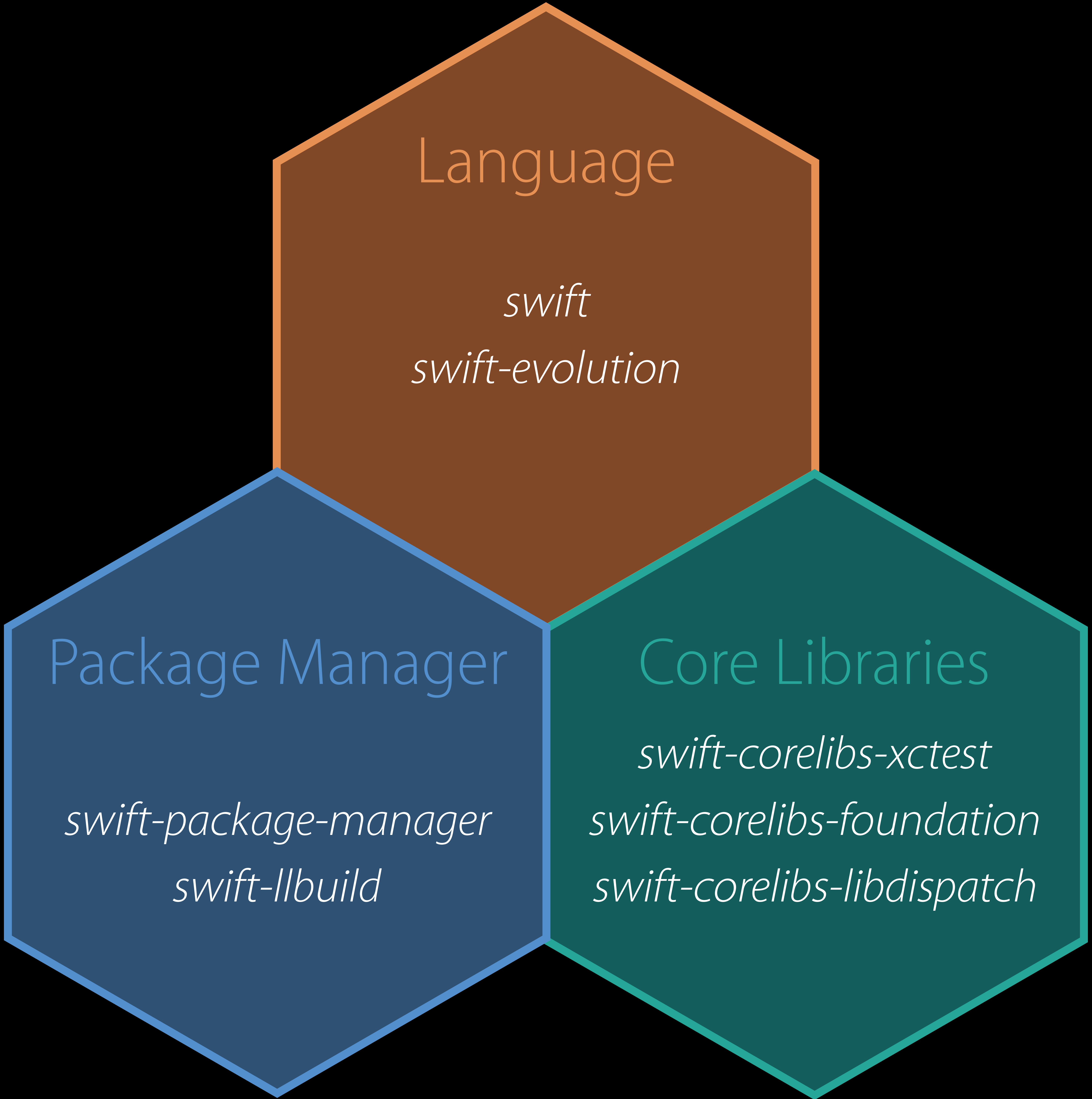
Decimal

DateComponents

IndexPath

DateInterval

IndexSet


Language
swift
swift-evolution
Package Manager
swift-package-manager
swift-llbuild
Core Libraries
swift-corelibs-xctest
swift-corelibs-foundation
swift-corelibs-libdispatch

Language

swift

swift-evolution

Language

swift

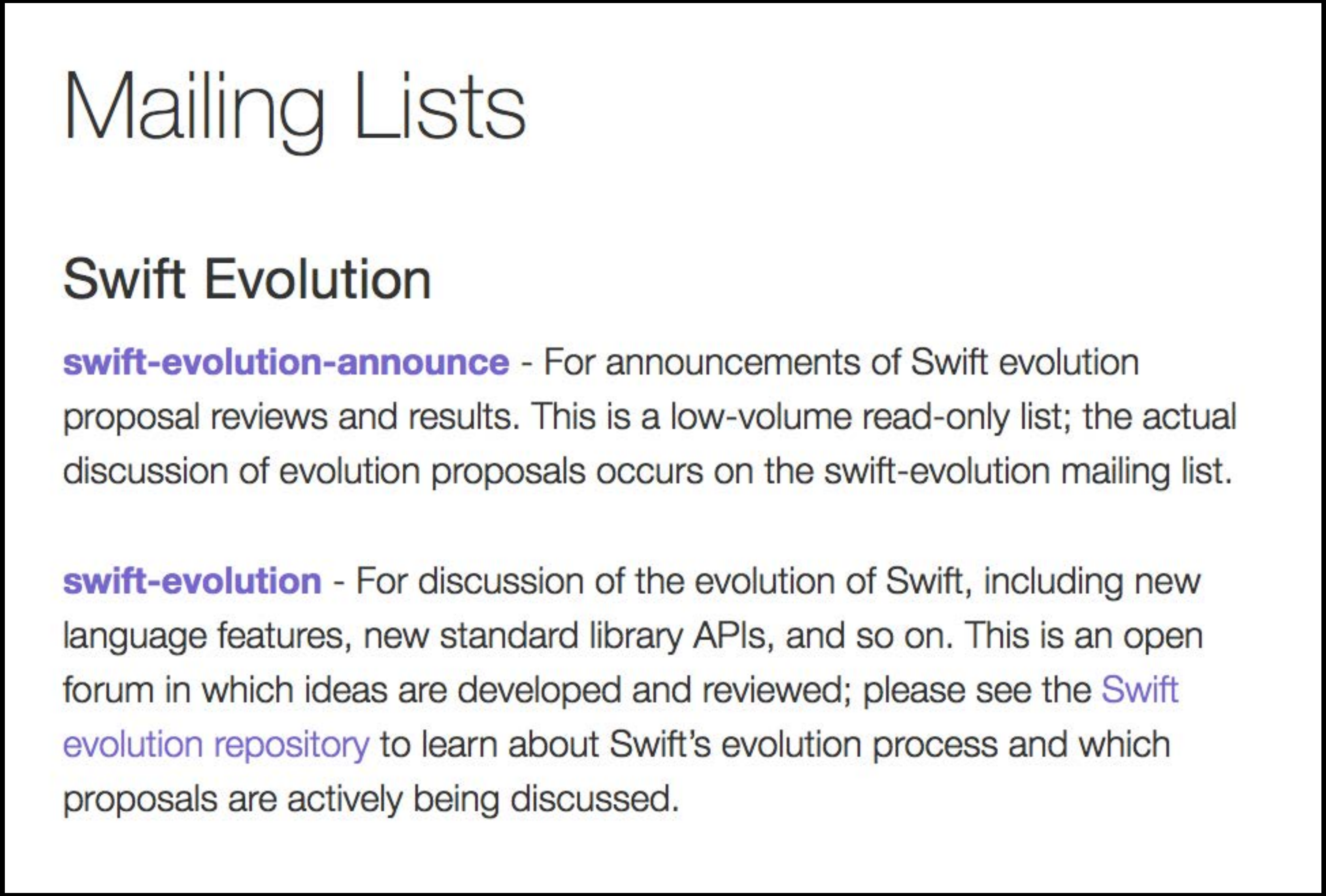
swift-evolution

swift-evolution

Language Evolution Process

Language Evolution Process

Socialize change on mailing list



Mailing Lists

Swift Evolution

swift-evolution-announce - For announcements of Swift evolution proposal reviews and results. This is a low-volume read-only list; the actual discussion of evolution proposals occurs on the swift-evolution mailing list.

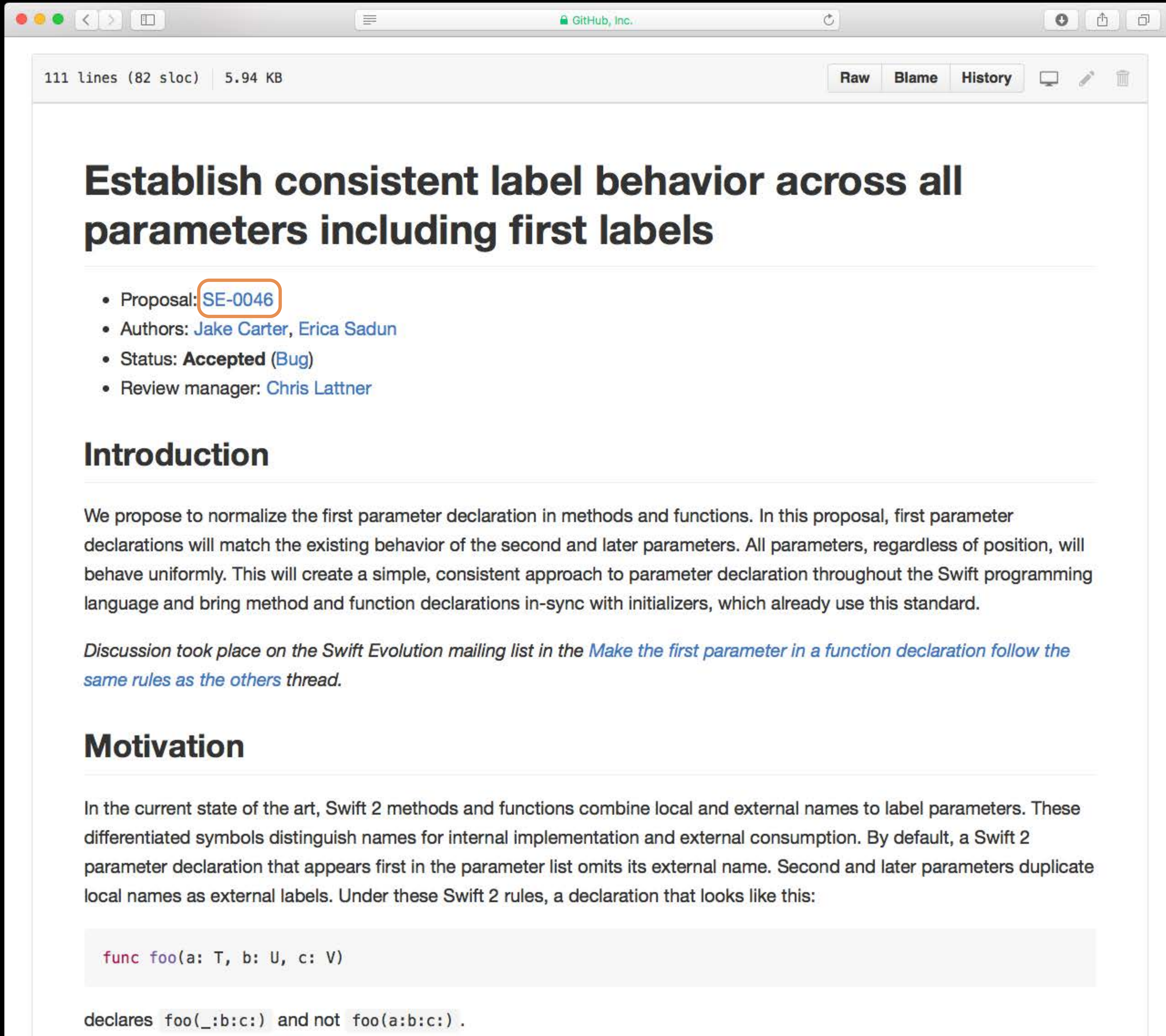
swift-evolution - For discussion of the evolution of Swift, including new language features, new standard library APIs, and so on. This is an open forum in which ideas are developed and reviewed; please see the Swift evolution repository to learn about Swift's evolution process and which proposals are actively being discussed.

Language Evolution Process

Socialize change on mailing list

Proposal submitted as a pull request

SE-0046



Establish consistent label behavior across all parameters including first labels

- Proposal: SE-0046
- Authors: Jake Carter, Erica Sadun
- Status: **Accepted** (Bug)
- Review manager: Chris Lattner

Introduction

We propose to normalize the first parameter declaration in methods and functions. In this proposal, first parameter declarations will match the existing behavior of the second and later parameters. All parameters, regardless of position, will behave uniformly. This will create a simple, consistent approach to parameter declaration throughout the Swift programming language and bring method and function declarations in-sync with initializers, which already use this standard.

Discussion took place on the Swift Evolution mailing list in the Make the first parameter in a function declaration follow the same rules as the others thread.

Motivation

In the current state of the art, Swift 2 methods and functions combine local and external names to label parameters. These differentiated symbols distinguish names for internal implementation and external consumption. By default, a Swift 2 parameter declaration that appears first in the parameter list omits its external name. Second and later parameters duplicate local names as external labels. Under these Swift 2 rules, a declaration that looks like this:

```
func foo(a: T, b: U, c: V)
```

declares `foo(_:b:c:)` and not `foo(a:b:c:)`.

Language Evolution Process

Socialize change on mailing list

Proposal submitted as a pull request

Pull request accepted to start review

Language Evolution Process

Socialize change on mailing list

Proposal submitted as a pull request

Pull request accepted to start review

Formal review on mailing lists

Language Evolution Process

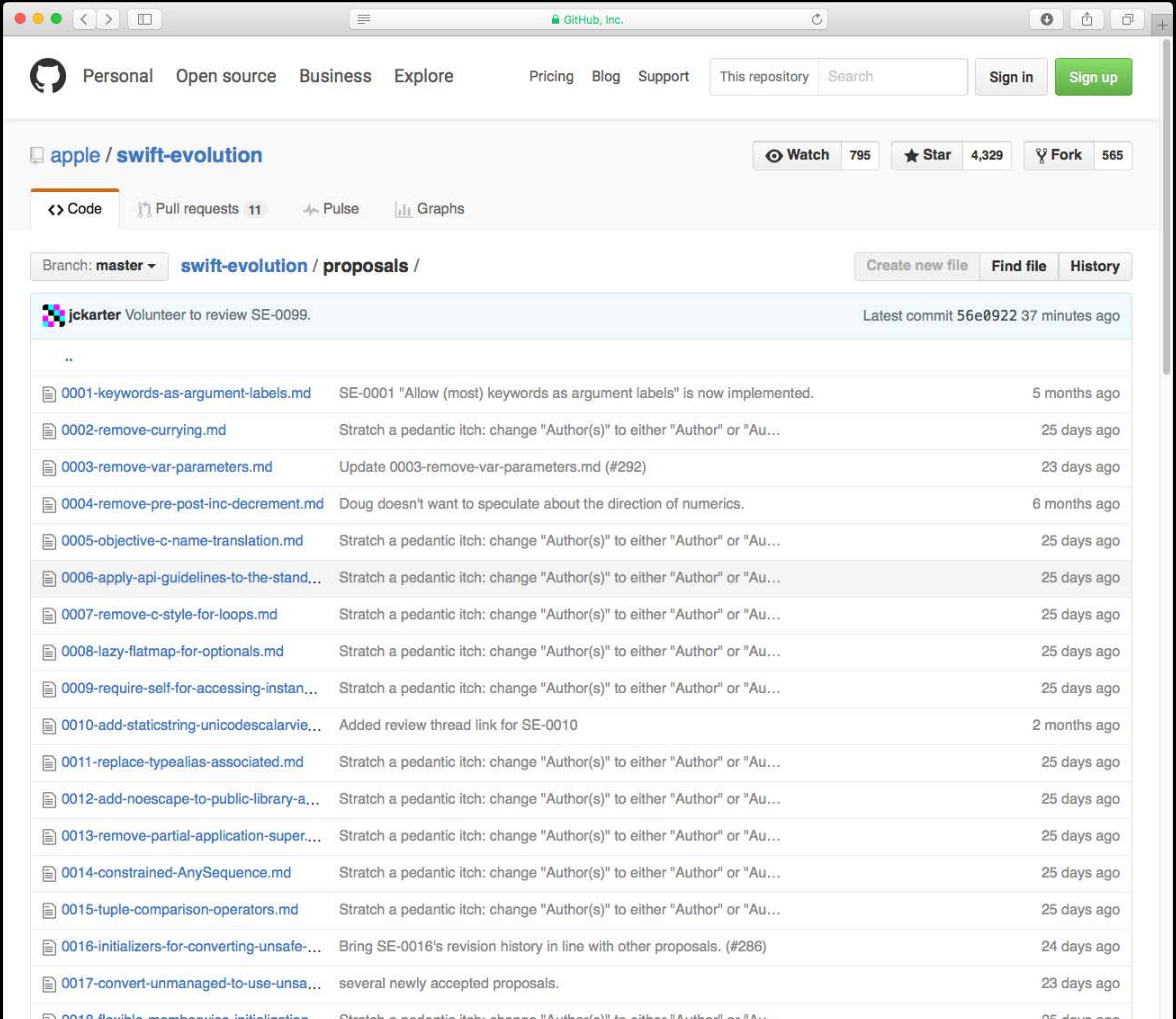
Socialize change on mailing list

Proposal submitted as a pull request

Pull request accepted to start review

Formal review on mailing lists

Core team arbitrates a decision



apple / swift-evolution

Branch: master   swift-evolution / proposals /

jckarter Volunteer to review SE-0099.   Latest commit 56e0922 37 minutes ago

..

0001-keywords-as-argument-labels.md	SE-0001 "Allow (most) keywords as argument labels" is now implemented.	5 months ago
0002-remove-currying.md	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0003-remove-var-parameters.md	Update 0003-remove-var-parameters.md (#292)	23 days ago
0004-remove-pre-post-inc-decrement.md	Doug doesn't want to speculate about the direction of numerics.	6 months ago
0005-objective-c-name-translation.md	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0006-apply-api-guidelines-to-the-stand…	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0007-remove-c-style-for-loops.md	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0008-lazy-flatmap-for-optionals.md	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0009-require-self-for-accessing-instan…	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0010-add-staticstring-unicodescalarvie…	Added review thread link for SE-0010	2 months ago
0011-replace-typealias-associated.md	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0012-add-noescape-to-public-library-a…	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0013-remove-partial-application-super…	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0014-constrained-AnySequence.md	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0015-tuple-comparison-operators.md	Stratch a pedantic itch: change "Author(s)" to either "Author" or "Au…	25 days ago
0016-initializers-for-converting-unsafe-…	Bring SE-0016's revision history in line with other proposals. (#286)	24 days ago
0017-convert-unmanaged-to-use-unsa…	several newly accepted proposals.	23 days ago

Language and Experience

Chris Lattner

Making the Core Experience Great

Improve overall experience of writing Swift code

- Swift language
- Standard library
- Cocoa in Swift
- Tools

Zeroing in on Source Compatibility

Primary goal of Swift 3

Source compatibility is the most popular “feature” request

Especially critical for cross-platform

Source compatibility between Swift 3 and 4 is a very **strong goal***

Zeroing in on Source Compatibility

Primary goal of Swift 3

Source compatibility is the most popular “feature” request

Especially critical for cross-platform

Source compatibility between Swift 3 and 4 is a very **strong goal***

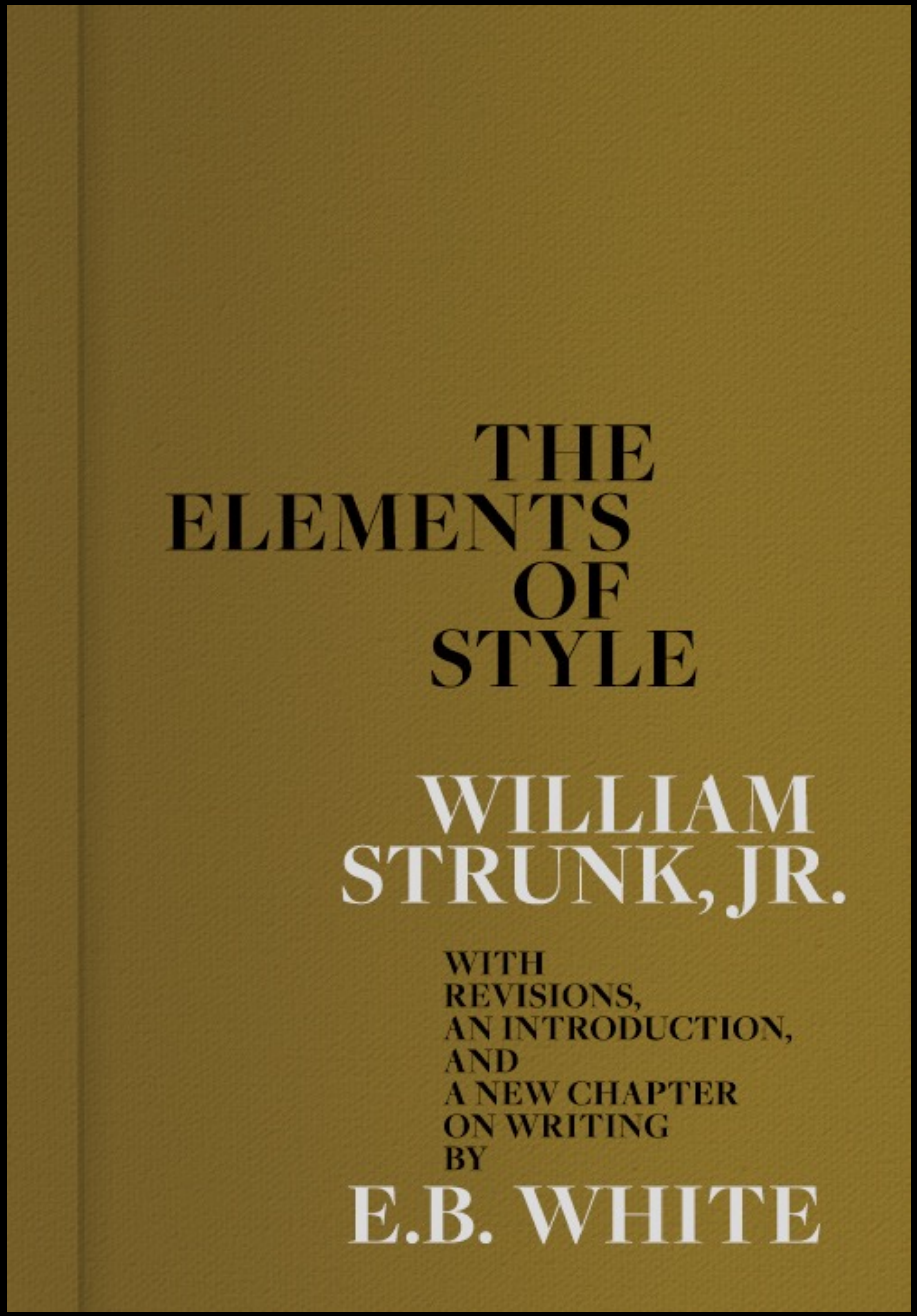
* But not an absolute promise

API Naming

Naming Guidelines

Carefully studied what is important in API design

- Strive for clarity—not terseness or verbosity
- Capture essential information
- Omit redundant information/boilerplate


THE ELEMENTS OF STYLE
WILLIAM STRUNK, JR.
WITH REVISIONS, AN INTRODUCTION, AND A NEW CHAPTER ON WRITING BY
E.B. WHITE

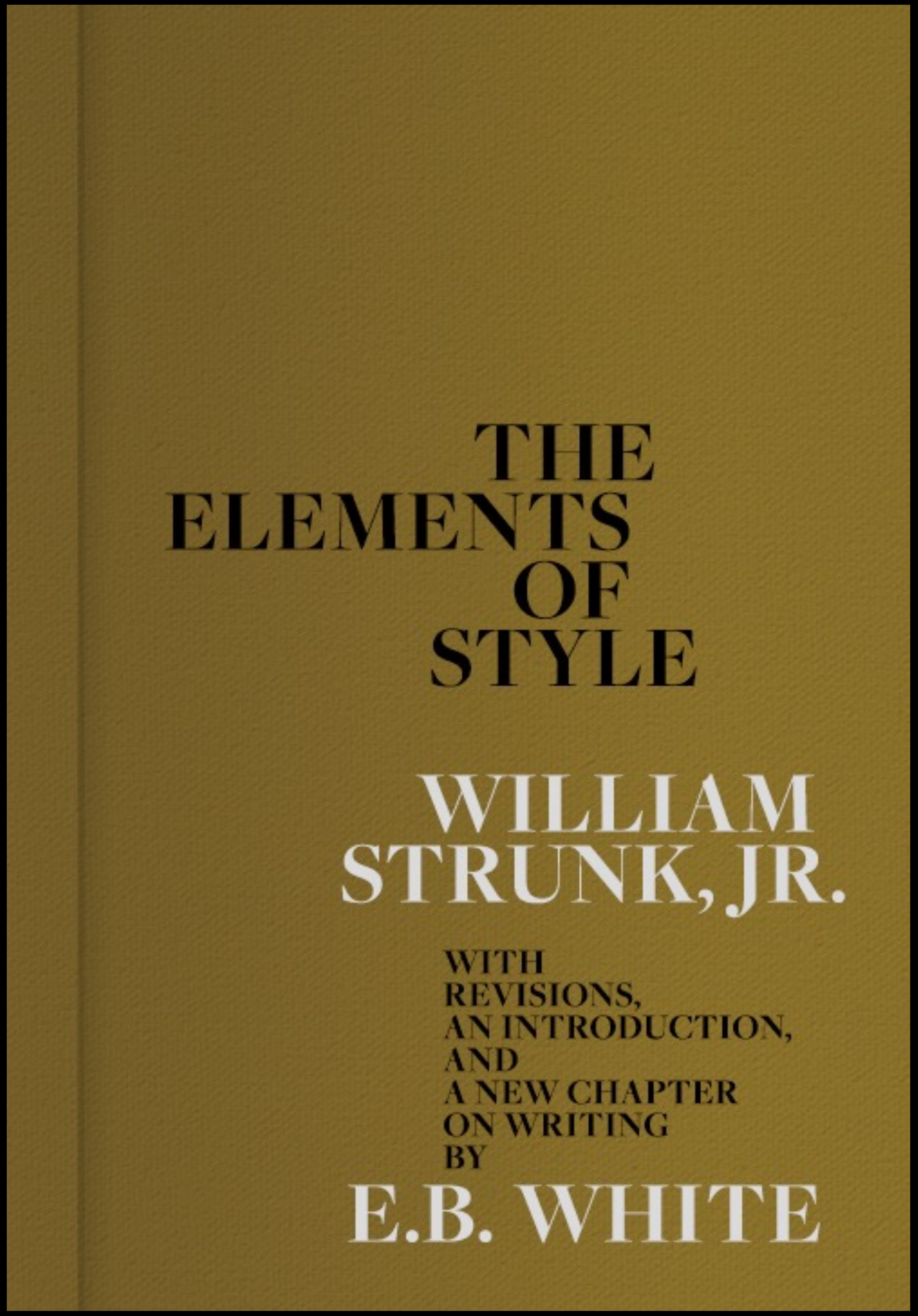
SE-0023 SE-0005 SE-0006

Naming Guidelines

Carefully studied what is important in API design

- Strive for clarity—not terseness or verbosity
- Capture essential information
- Omit redundant information/boilerplate

https://swift.org/documentation/api-design-guidelines/


THE ELEMENTS OF STYLE
WILLIAM STRUNK, JR.
WITH REVISIONS, AN INTRODUCTION, AND A NEW CHAPTER ON WRITING BY
E.B. WHITE

SE-0023 SE-0005 SE-0006

Example API Changes

```swift
array.appendContentsOf([2,3,4])
array.insert(1, atIndex: 0)
```

Swift.Array

Example API Changes

```
array.appendContentsOf([2,3,4])
array.insert(1, atIndex: 0)
```

Swift.Array

```
if url.fileURL {}
x = url.URLByAppendingPathComponent("file.txt")
```

Foundation.NSURL

Example API Changes

```swift
array.append(contentsOf: [2,3,4])
array.insert(1, atIndex: 0)
```

Swift.Array

```swift
if url.fileURL {}
x = url.URLByAppendingPathComponent("file.txt")
```

Foundation.NSURL

Example API Changes

```
array.append(contentsOf: [2,3,4])
array.insert(1, at: 0)
```

Swift.Array

```
if url.fileURL {}
x = url.URLByAppendingPathComponent("file.txt")
```

Foundation.NSURL

Example API Changes

```
array.append(contentsOf: [2,3,4])
array.insert(1, at: 0)
```

Swift.Array

```
if url.isFileURL {}
x = url.URLByAppendingPathComponent("file.txt")
```

Foundation.NSURL

Example API Changes

```swift
array.append(contentsOf: [2,3,4])
array.insert(1, at: 0)
```

Swift.Array

```swift
if url.isFileURL {}
x = url.appendingPathComponent("file.txt")
```

Foundation.NSURL

Example API Changes

```
array.append(contentsOf: [2,3,4])
array.insert(1, at: 0)
```

Swift.Array

```
if url.isFileURL {}
x = url.appendingPathComponent("file.txt")
```

Foundation.NSURL

Swift API Design Guidelines Presidio Tuesday 10:00AM

Importing Objective-C APIs

Import as Member

```
void CGContextFillPath(CGContextRef);
```

SE-0044

Import as Member

```
void CGContextFillPath(CGContextRef);
```

```
func CGContextFillPath(_: CGContext)
```

Swift 2

SE-0044

Import as Member

```
void CGContextFillPath(CGContextRef)
           NS_SWIFT_NAME(CGContext.fillPath(self:));
```

```
func CGContextFillPath(_: CGContext)
```

Swift 2

```
extension CGContext {
  func fillPath()
}
```

Swift 3

Swift API Design Guidelines   Presidio   Tuesday 10:00AM

SE-0044

Objective-C Generics

```
func findAnimals() {
  let request = NSFetchRequest(entityName:”Animal")
  guard let searchResults =
           try? context.executeFetchRequest(request) as! [Animal] {
    return
  }
  ...
  use(searchResults)
}
```

SE-0057

Objective-C Generics

```
func findAnimals() {
  let request : NSFetchRequest<Animal> = Animal.fetchRequest
  guard let searchResults = try? context.fetch(request) {

    return
  }
  ...
  use(searchResults)
}
```

SE-0057

Stringly Typed Objective-C Constants

```
typedef NSString *NSNotificationName;
const NSNotificationName NSUserDefaultsDidChangeNotification;
```

SE-0033

Stringly Typed Objective-C Constants

```
typedef NSString *NSNotificationName;
const NSNotificationName NSUserDefaultsDidChangeNotification;
```

Imported definition

```
  let NSUserDefaultsDidChangeNotification: String
```

SE-0033

Stringly Typed Objective-C Constants

```
typedef NSString *NSNotificationName;
const NSNotificationName NSUserDefaultsDidChangeNotification;
```

Imported definition

```
  let NSUserDefaultsDidChangeNotification: String
```

Use

```
  center.addObserver(forName: NSUserDefaultsDidChangeNotification, …)
```

SE-0033

Stringly Typed Objective-C Constants

```
typedef NSString *NSNotificationName NS_EXTENSIBLE_STRING_ENUM;
const NSNotificationName NSUserDefaultsDidChangeNotification;
```

Imported definition

```
  let NSUserDefaultsdidChangeNotification: String
```

Use

```
  center.addObserver(forName: NSUserDefaultsDidChangeNotification, …)
```

SE-0033

Stringly Typed Objective-C Constants

```
typedef NSString *NSNotificationName NS_EXTENSIBLE_STRING_ENUM;
const NSNotificationName NSUserDefaultsDidChangeNotification;
```

Imported definition

```
extension UserDefaults {
  class let didChangeNotification: NSNotification.Name
}
```

Use

```
center.addObserver(forName: NSUserDefaultsDidChangeNotification, …)
```

SE-0033

Stringly Typed Objective-C Constants

```
typedef NSString *NSNotificationName NS_EXTENSIBLE_STRING_ENUM;
const NSNotificationName NSUserDefaultsDidChangeNotification;
```

Imported definition

```
extension UserDefaults {
  class let didChangeNotification: NSNotification.Name
}
```

Use

```
center.addObserver(forName: NSUserDefaultsdidChangeNotification, …)
```

SE-0033

Stringly Typed Objective-C Constants

```
typedef NSString *NSNotificationName NS_EXTENSIBLE_STRING_ENUM;
const NSNotificationName NSUserDefaultsDidChangeNotification;
```

Imported definition

```
extension UserDefaults {
  class let didChangeNotification: NSNotification.Name
}
```

Use

```
center.addObserver(forName: UserDefaults.didChangeNotification, …)
```

SE-0033

Strongly Typed Objective-C Constants

```
typedef NSString *NSNotificationName NS_EXTENSIBLE_STRING_ENUM;
const NSNotificationName NSUserDefaultsDidChangeNotification;
```

Imported definition

```
extension UserDefaults {
  class let didChangeNotification: NSNotification.Name
}
```

Use

```
center.addObserver(forName: UserDefaults.didChangeNotification, …)
```

SE-0033

Improvements Throughout the SDK

Major work on Foundation, Dispatch, and Core Graphics

Countless smaller improvements

- Ongoing nullability audit
- Adoption of Objective-C generics
- And more…

Improvements Throughout the SDK

Major work on Foundation, Dispatch, and Core Graphics

Countless smaller improvements

- Ongoing nullability audit
- Adoption of Objective-C generics
- And more…

What's New in Foundation for Swift	Mission	Tuesday 4:00PM
Concurrent Programming with GCD in Swift 3	Pacific Heights	Friday 4:00PM

Core Language

Consistent Parameter Labeling

```swift
func myFunction(a: Int, b: Int, c: Int) { }
```

SE-0046

Consistent Parameter Labeling

```swift
func myFunction(a: Int, b: Int, c: Int) { }

myFunction(42, b: 57, c: 99)
```

Swift 2

SE-0046

Consistent Parameter Labeling

```
func myFunction(a: Int, b: Int, c: Int) { }

    myFunction(a: 42, b: 57, c: 99)
```

Swift 3

SE-0046

Consistent Parameter Labeling

```
func myFunction(a: Int, b: Int, c: Int) { }

      myFunction(a: 42, b: 57, c: 99)
```

Swift 3

Simpler and more consistent

API naming often encourages first parameter label

Any parameter label may be suppressed with _

SE-0046

Move ‘where’ Clause to End of Declaration

COMING SOON!

```
func anyCommon<T: Sequence, U: Sequence>(lhs: T, rhs: U) -> Bool {
```

SE-0081

Move ‘where’ Clause to End of Declaration

COMING SOON!

```
func anyCommon<T: Sequence, U: Sequence
               where T.Element: Equatable,
                     T.Element == U.Element
               >(lhs: T, rhs: U) -> Bool {
```

SE-0081

Move 'where' Clause to End of Declaration

COMING SOON!

```
func anyCommon<T: Sequence, U: Sequence
               where T.Element: Equatable,
                     T.Element == U.Element
               >(lhs: T, rhs: U) -> Bool {
```

Swift 2

SE-0081

Move 'where' Clause to End of Declaration

COMING SOON!

```
func anyCommon<T: Sequence, U: Sequence
               where T.Element: Equatable,
                     T.Element == U.Element
               >(lhs: T, rhs: U) -> Bool {
```

Swift 2

```
func anyCommon<T: Sequence, U: Sequence>(lhs: T, rhs: U) -> Bool
    where T.Element: Equatable, T.Element == U.Element {
```

Swift 3

SE-0081

Warn on Unused Results by Default

```
func plusOne(_ a: Int) -> Int {
  return a+1
}



plusOne(x)
```

SE-0047

Warn on Unused Results by Default

```
func plusOne(_ a: Int) -> Int {
   return a+1
}



plusOne(x)
```

⚠ Result of call to ‘plusOne’ is unused

SE-0047

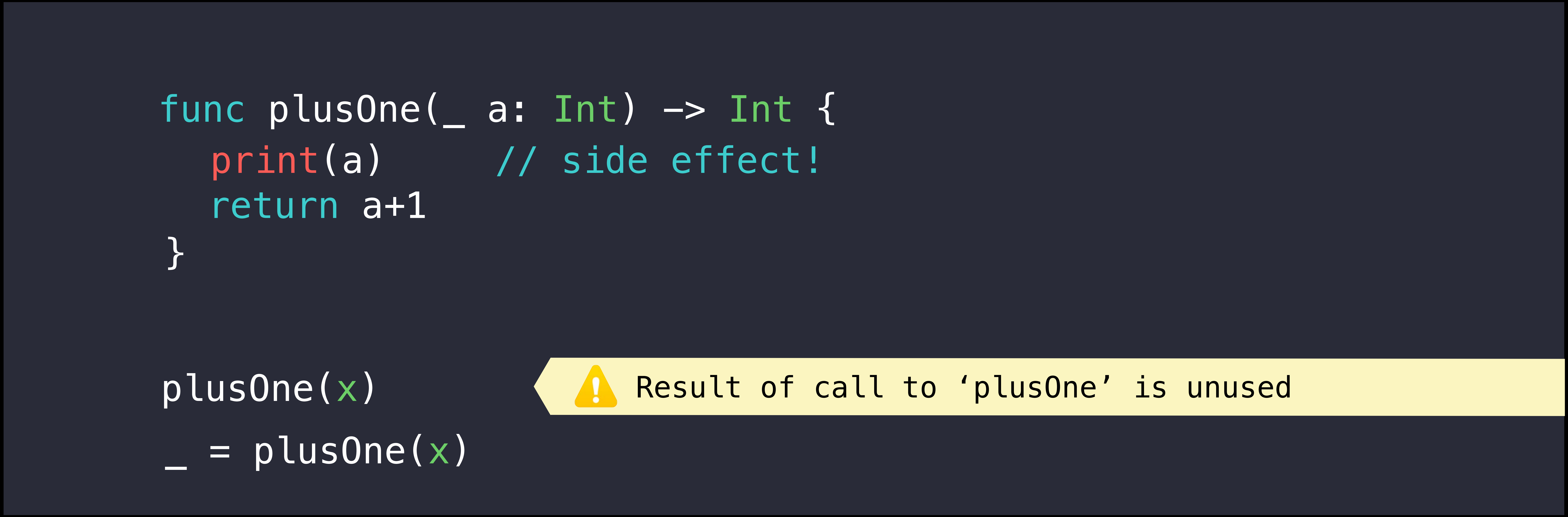
Warn on Unused Results by Default

```swift
func plusOne(_ a: Int) -> Int {
   print(a)     // side effect!
   return a+1
}


plusOne(x)          ⚠ Result of call to ‘plusOne’ is unused
```

SE-0047

Warn on Unused Results by Default



```
func plusOne(_ a: Int) -> Int {
   print(a)      // side effect!
   return a+1
}


plusOne(x)           Result of call to ‘plusOne’ is unused
_ = plusOne(x)
```

SE-0047

Warn on Unused Results by Default

```swift
@discardableResult
func plusOne(_ a: Int) -> Int {
   print(a)     // side effect!
   return a+1
}


plusOne(x)

_ = plusOne(x)
```

SE-0047

Features Removed in Swift 3

Features Removed in Swift 3

Features Removed in Swift 3

Focus and simplify the language

Reduce language complexity

Teaching and learning

Features Removed in Swift 3

Focus and simplify the language

Reduce language complexity

Teaching and learning

What got removed?

SE-0002	Currying func declaration syntax
SE-0003	`var` in function parameter lists
SE-0004	`++` and `--` operators
SE-0007	C-style `for` loop
SE-0029	Implicit tuple splat in calls

Core Language

Other small enhancements

Core Language

Other small enhancements

Core Language

Other small enhancements

COMING SOON!

SE-0025	Scoped access level, new `fileprivate` access level
SE-0043	`case` labels with multiple variable bindings
SE-0048	Generic Type Aliases
SE-0062	Referencing Objective-C key-paths
SE-0064	Referencing the selector for property getters and setters
SE-0068	Expanding `Self` to class members and value types
SE-0075	Adding a build configuration "is importable" test
SE-0092	Typealiases in protocols and protocol extensions

Core Language

Syntactic cleanups

Core Language

Syntactic cleanups

Core Language

Syntactic cleanups

SE-0028	Replace `__FILE__` with `#file`
SE-0031	`inout` moved to be part of the type
SE-0036	Requiring leading dot prefixes for enum instance members
SE-0040	Attribute syntax: replace `=` with `:`
SE-0049	Move `@noescape` and `@autoclosure` to be type attributes
SE-0060	Enforcing order of defaulted parameters
SE-0066	Standardize function type argument syntax to require parentheses
SE-0096	Converting `dynamicType` from a property to an operator

Type System

Type System Purpose

Type system and type checker work together

- Validate correctness of code
- Infer types and overloads implicit in code

```
let a = x + y
```

Type System Purpose

Type system and type checker work together

- Validate correctness of code
- Infer types and overloads implicit in code

```
let a = x + y
```

Goal

- Simpler, more consistent, and more predictable type system
- Remove "gotchas" and surprising behavior
- Improve type checker performance

UnsafePointer Nullability

```
let ptr : UnsafeMutablePointer<Int> = nil

if ptr != nil {
   ptr.memory = 42
}
```

Swift 2

SE-0055

UnsafePointer Nullability

```
let ptr : UnsafeMutablePointer<Int>? = nil

if let ptr = ptr {
   ptr.memory = 42
}
```

Swift 3

SE-0055

UnsafePointer Nullability

```
let ptr : UnsafeMutablePointer<Int>? = nil

ptr .memory = 42
```

Swift 3

Imported C pointers in APIs obey `_Nullable` and `_Null_unspecified`

Consistency: `nil` is dedicated to `Optional` and `ImplicitlyUnwrappedOptional`

SE-0002

UnsafePointer Nullability

```
let ptr : UnsafeMutablePointer<Int>? = nil

ptr?.memory = 42
```

Swift 3

Imported C pointers in APIs obey `_Nullable` and `_Null_unspecified`

Consistency: `nil` is dedicated to `Optional` and `ImplicitlyUnwrappedOptional`

SE-0002

Implicitly Unwrapped Optional (IUO)

```
func f(value : Int!) {





}
```

Swift 2

SE-0054

Implicitly Unwrapped Optional (IUO)

```
func f(value : Int!) {
  let x = value + 1
  let y = value




}
```

Swift 2

SE-0054

Implicitly Unwrapped Optional (IUO)

```
func f(value : Int!) {
  let x = value + 1            // x: Int - force unwrapped
  let y = value                // y: Int!




}
```

Swift 2

SE-0054

Implicitly Unwrapped Optional (IUO)

```
func f(value : Int!) {
  let x = value + 1            // x: Int — force unwrapped
  let y = value                // y: Int!

  let array = [value, 42]     // [Int], [Int!], [Int?], [Any]...

  use(array)
}
```

Swift 2

SE-0054

Implicitly Unwrapped Optional (IUO)

```
func f(value : Int!) {
  let x = value + 1              // x: Int - force unwrapped
  let y = value                  // y: Int!

  let array = [value, 42]    // [Int], [Int!], [Int?], [Any]...

  use(array)   ! Cannot convert value of type '[Int!]' to argument type
}
```

Swift 2

SE-0054

Implicitly Unwrapped Optional (IUO)

```swift
func f(value : Int!) {
  let x = value + 1
  let y = value

  let array = [value, 42]

  use(array)
}
```

Swift 3

"IUO" becomes a strong optional if that will work

- It is only forced if necessary to type check

SE-0054

Implicitly Unwrapped Optional (IUO)

```swift
func f(value : Int!) {
  let x = value + 1              // x: Int – force unwrapped
  let y = value

  let array = [value, 42]

  use(array)
}
```

Swift 3

"IUO" becomes a strong optional if that will work

- It is only forced if necessary to type check

SE-0054

Implicitly Unwrapped Optional (IUO)

```
func f(value : Int!) {
  let x = value + 1             // x: Int – force unwrapped
  let y = value                 // y: Int?

  let array = [value, 42]

  use(array)
}
```

Swift 3

"IUO" becomes a strong optional if that will work

- It is only forced if necessary to type check

SE-0054

Implicitly Unwrapped Optional (IUO)

```
func f(value : Int!) {
  let x = value + 1          // x: Int – force unwrapped
  let y = value              // y: Int?

  let array = [value, 42]    // [Int?]

  use(array)
}
```

Swift 3

"IUO" becomes a strong optional if that will work

- It is only forced if necessary to type check

SE-0054

Implicitly Unwrapped Optional (IUO)

```
func f(value : Int!) {
  let x = value + 1           // x: Int – force unwrapped
  let y = value               // y: Int?

  let array = [value, 42]    // [Int?]
  let array2 = [value!, 42] // [Int]
  use(array)
}
```

Swift 3

“IUO” becomes a strong optional if that will work

- It is only forced if necessary to type check

SE-0054

Standard Library

New Collection Indexing Model

Collections move their indices

```
i = collection.startIndex

next = i.successor()
```

Swift 2

SE-0065

New Collection Indexing Model

Collections move their indices

```
i = collection.startIndex

next = i.successor()
```

Swift 2

```
i = collection.startIndex

next = collection.index(after: i)
```

Swift 3

SE-0065

New Collection Indexing Model

Collections move their indices

```
i = collection.startIndex

next = i.successor()
```

Swift 2

```
i = collection.startIndex

next = collection.index(after: i)
```

Swift 3

Benefits

- `HalfOpenInterval` and `IntervalType` are merged into `Range`
- `0...UInt8.max` now works properly
- Better performance

SE-0065

Floating Point and Numerics

New `FloatingPoint` protocol unifies `Float`, `Double`, `Float80`, and `CGFloat`

- Provides core IEEE-754 properties and operations
- Permits algorithms to be generic over all floating point types

SE-0067

Floating Point and Numerics

New `FloatingPoint` protocol unifies `Float`, `Double`, `Float80`, and `CGFloat`

- Provides core IEEE-754 properties and operations
- Permits algorithms to be generic over all floating point types

```
let v = 2 * Float(M_PI)
```

Swift 2

```
let v = 2 * Float.pi
```

Swift 3

SE-0067

Floating Point and Numerics

New `FloatingPoint` protocol unifies `Float`, `Double`, `Float80`, and `CGFloat`

- Provides core IEEE-754 properties and operations
- Permits algorithms to be generic over all floating point types

```swift
let v = 2 * Float(M_PI)

return x * CGFloat(M_PI) / 180
```

Swift 2

```swift
let v = 2 * Float.pi

return x * CGFloat.pi / 180
```

Swift 3

SE-0067

Floating Point and Numerics

New `FloatingPoint` protocol unifies `Float`, `Double`, `Float80`, and `CGFloat`

- Provides core IEEE-754 properties and operations
- Permits algorithms to be generic over all floating point types

```swift
   let v = 2 * Float(M_PI)

return x * CGFloat(M_PI) / 180
```

Swift 2

```swift
   let v = 2 * Float.pi

return x * .pi / 180
```

Swift 3

SE-0067

Standard Library

Other small enhancements

Standard Library

Other small enhancements

SE-0008	Add a Lazy `flatMap` for sequences of optionals
SE-0016	Conversions `Unsafe[Mutable]Pointer` to `Int` and `UInt`
SE-0017	Change `Unmanaged` to use `UnsafePointer`
SE-0032	Add `first(where:)` method to `Sequence`
SE-0061	Add generic result and error handling to `autoreleasepool()`
SE-0080	Failable numeric conversion initializers
SE-0093	Adding a public `base` property to slices
SE-0094	Add `sequence(first:next:)` and `sequence(state:next:)` to the stdlib

Swift 3 Language and Experience

API naming

Importing Objective-C APIs

Core language

Type system

Standard library

Swift Tools

Ewa Matejska

Whole Module Optimization

Faster Type Checking

String Hashing

Faster Startup Time

Faster Dictionary

Smaller Binaries

Incremental Compilation

Stack Promotion

3x

Dictionary<String, T>

24x

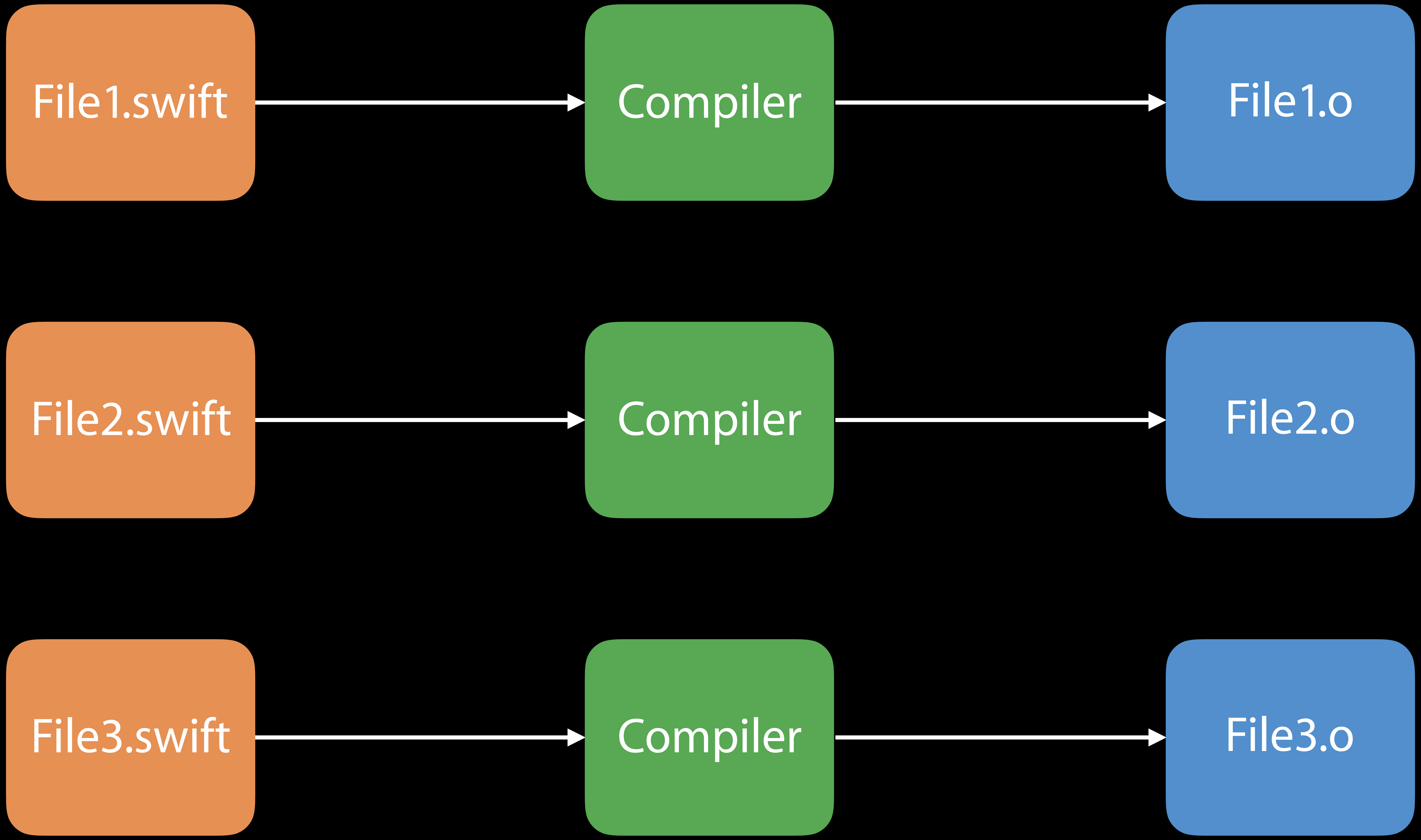
Heap to Stack Promotion for Classes

86x

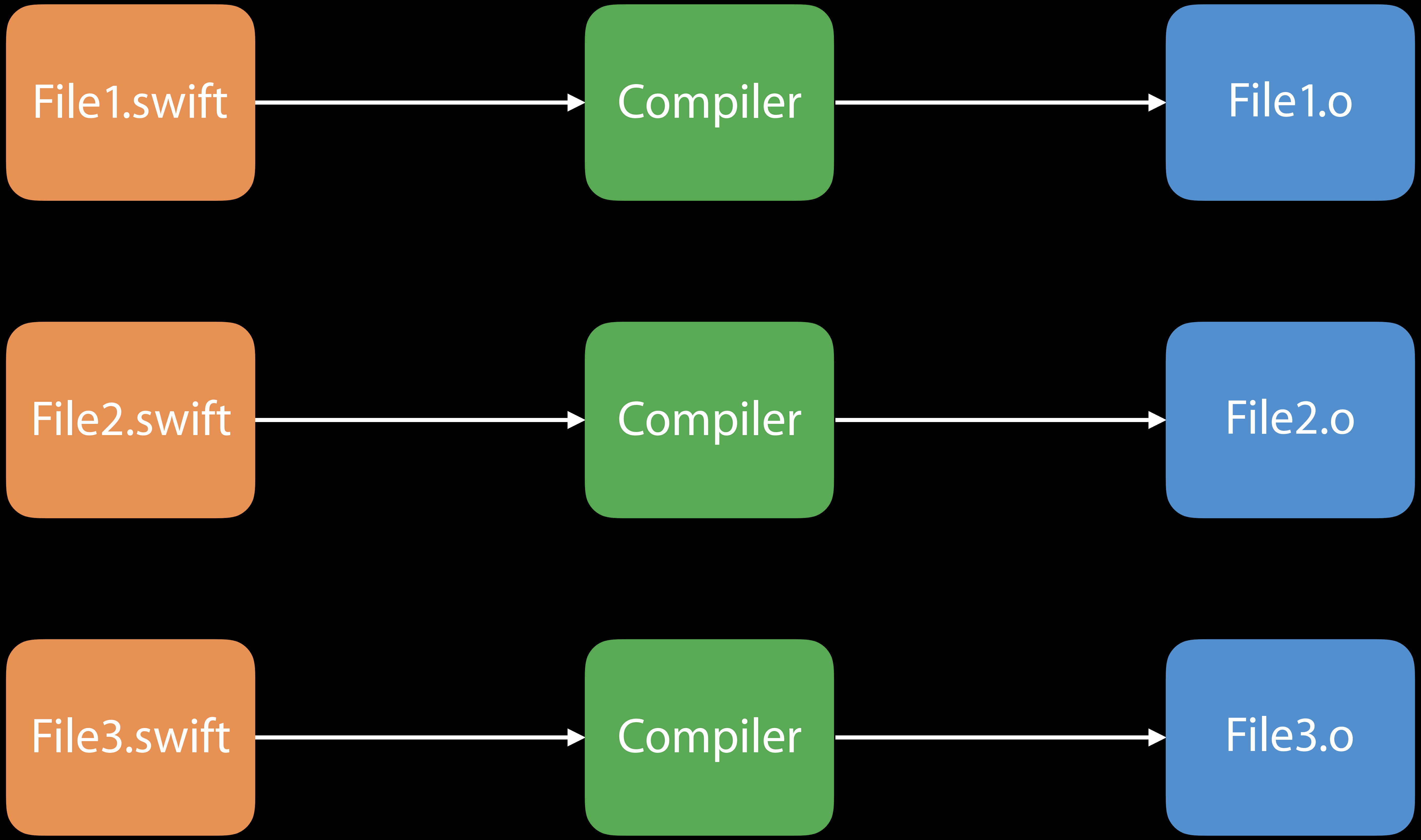
String Algorithm Optimizations

Whole Module Optimization

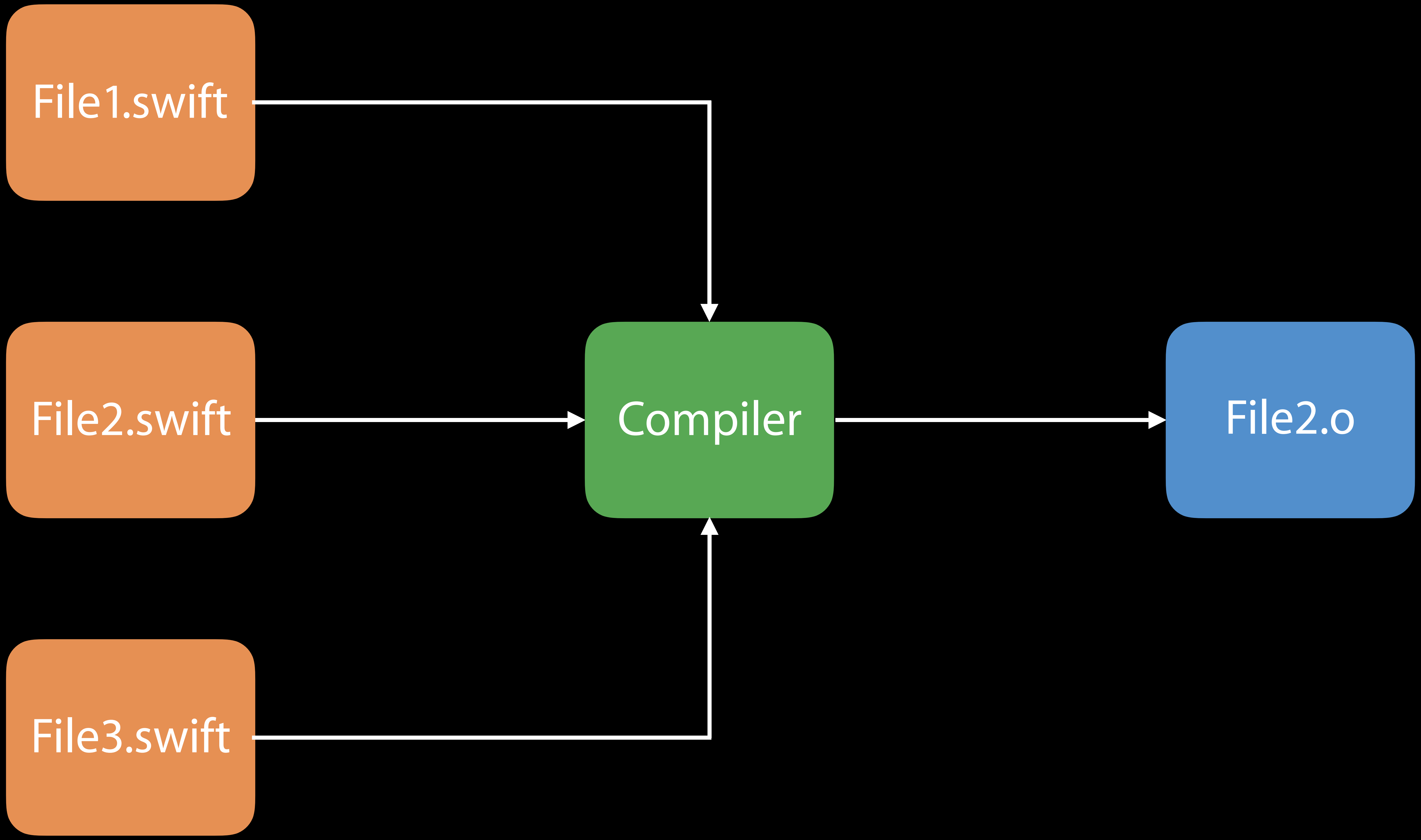
Whole Module Optimization



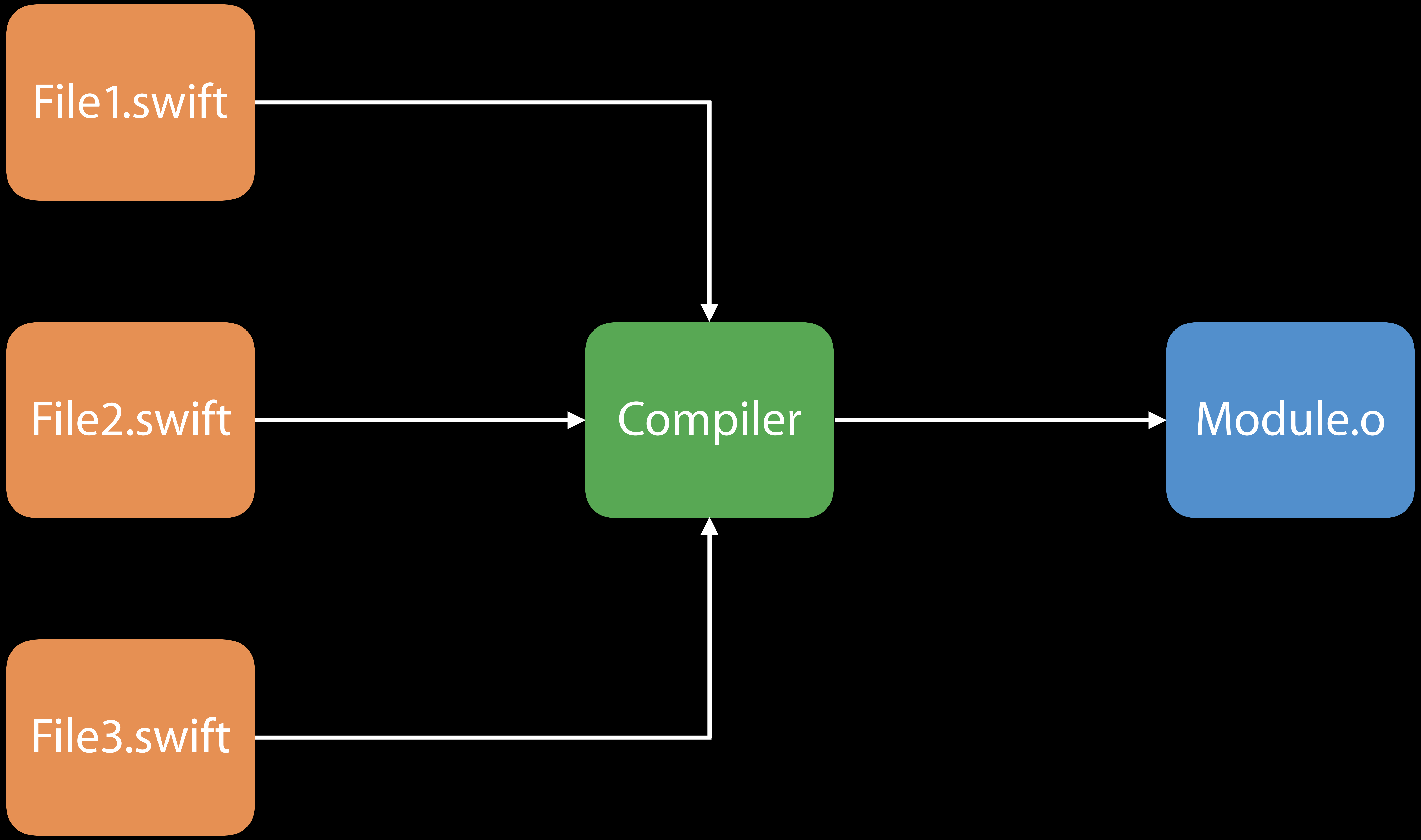
Whole Module Optimization



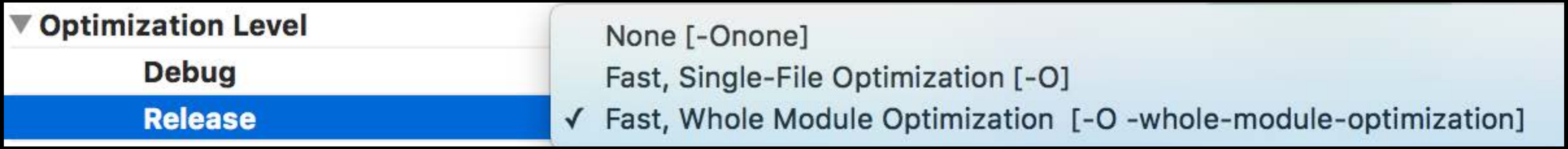
Whole Module Optimization



Whole Module Optimization



WMO on by Default for New Projects



▼ Optimization Level

Debug

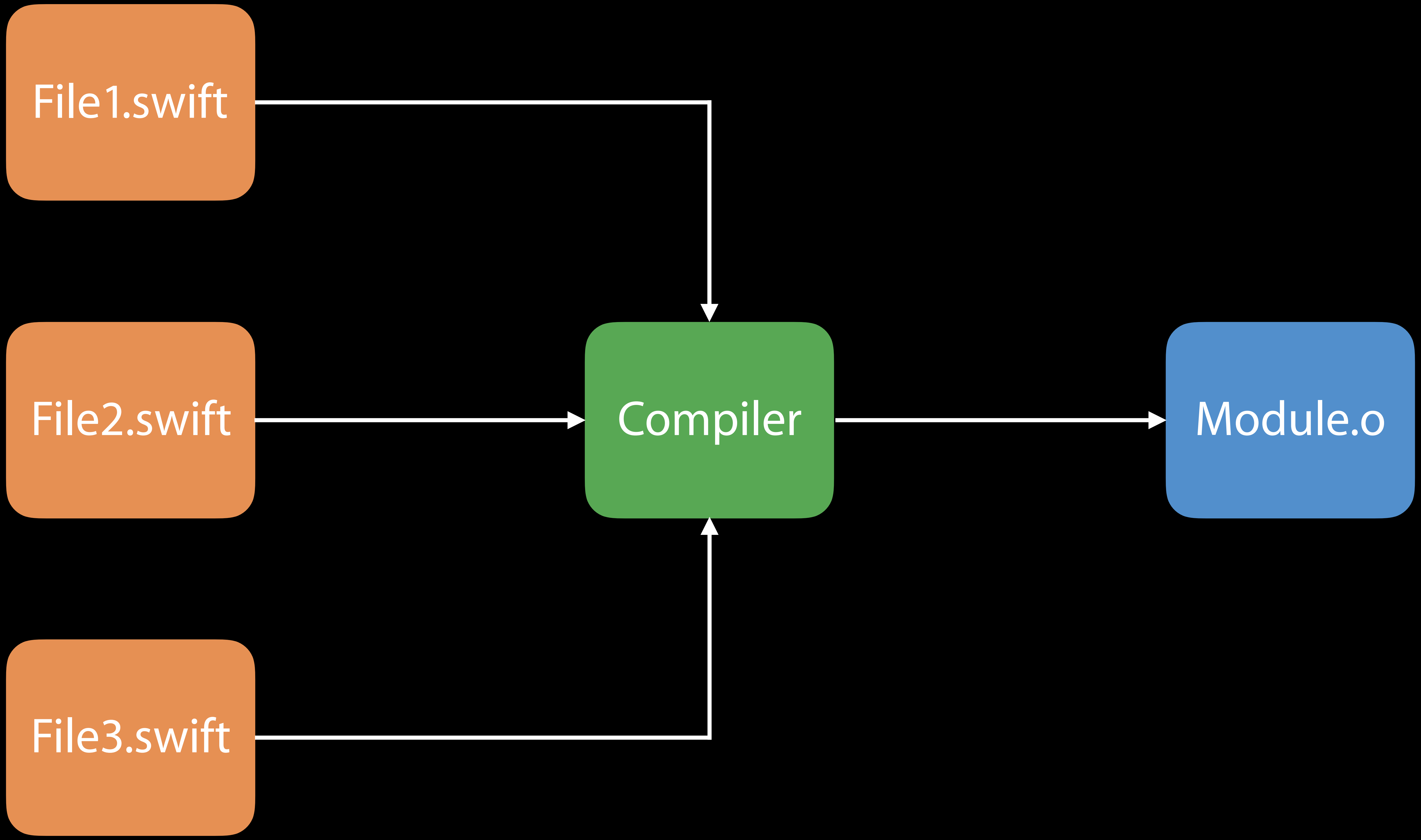
Release

None [-Onone]

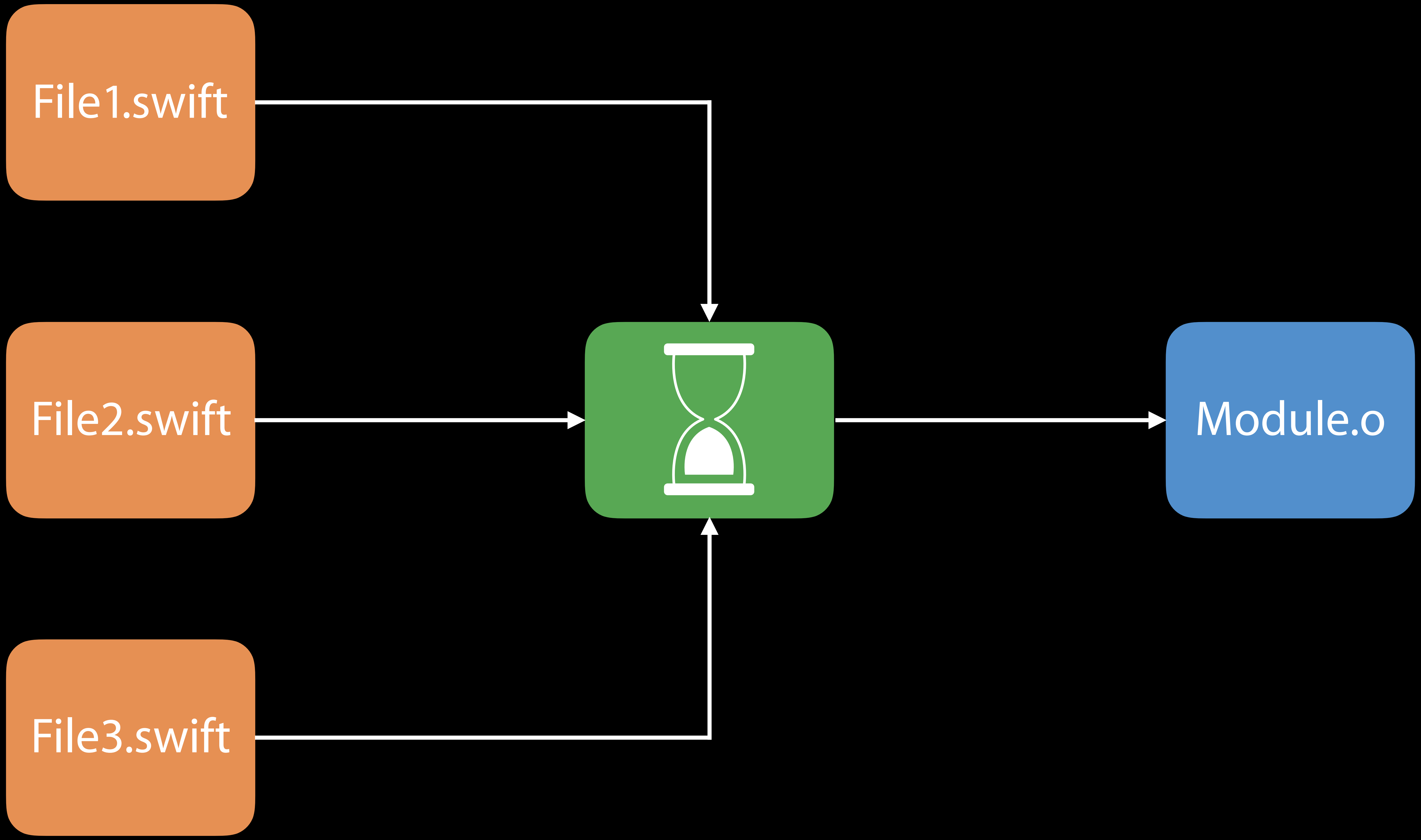
Fast, Single-File Optimization [-O]

✓ Fast, Whole Module Optimization [-O -whole-module-optimization]

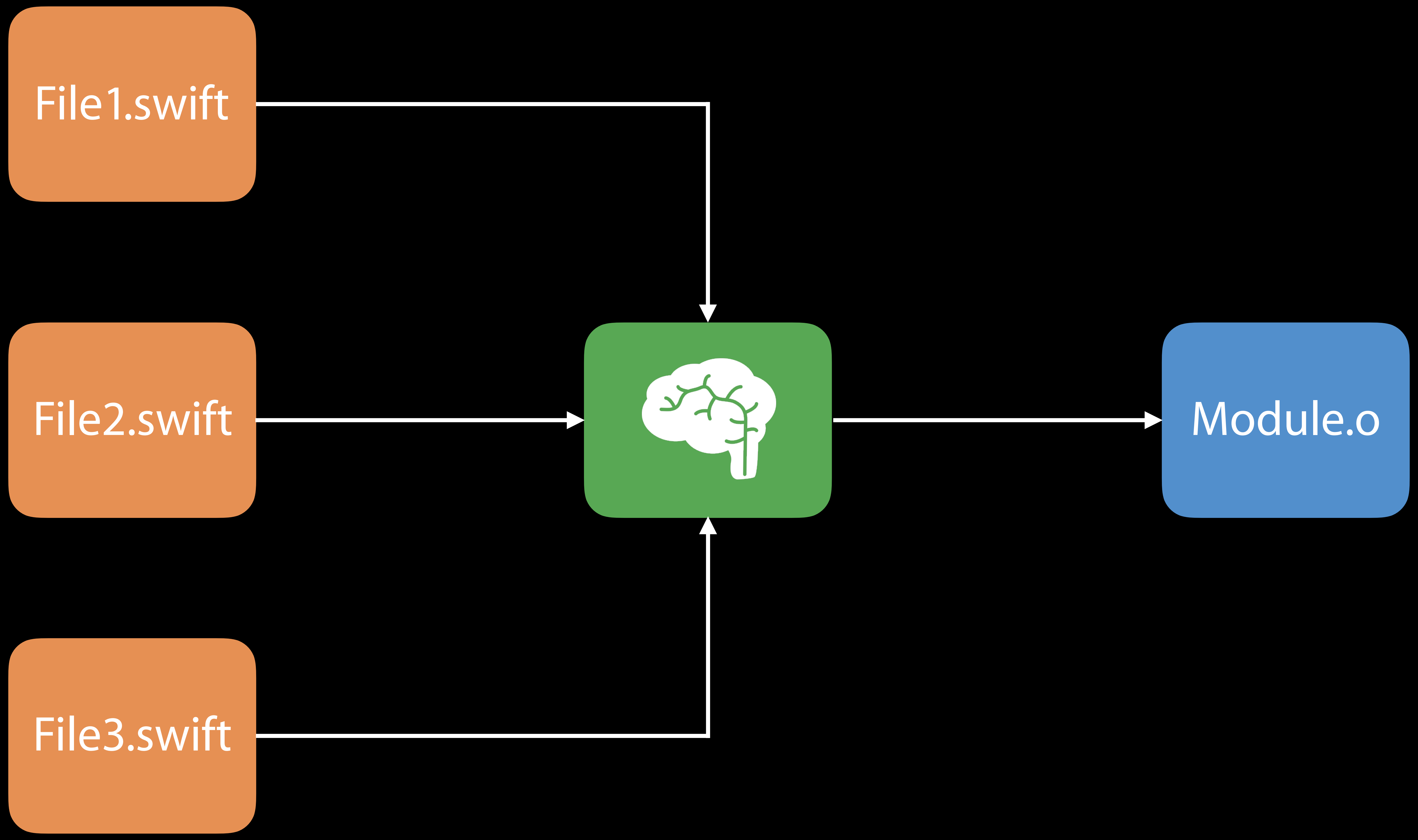
What About Compile Time?



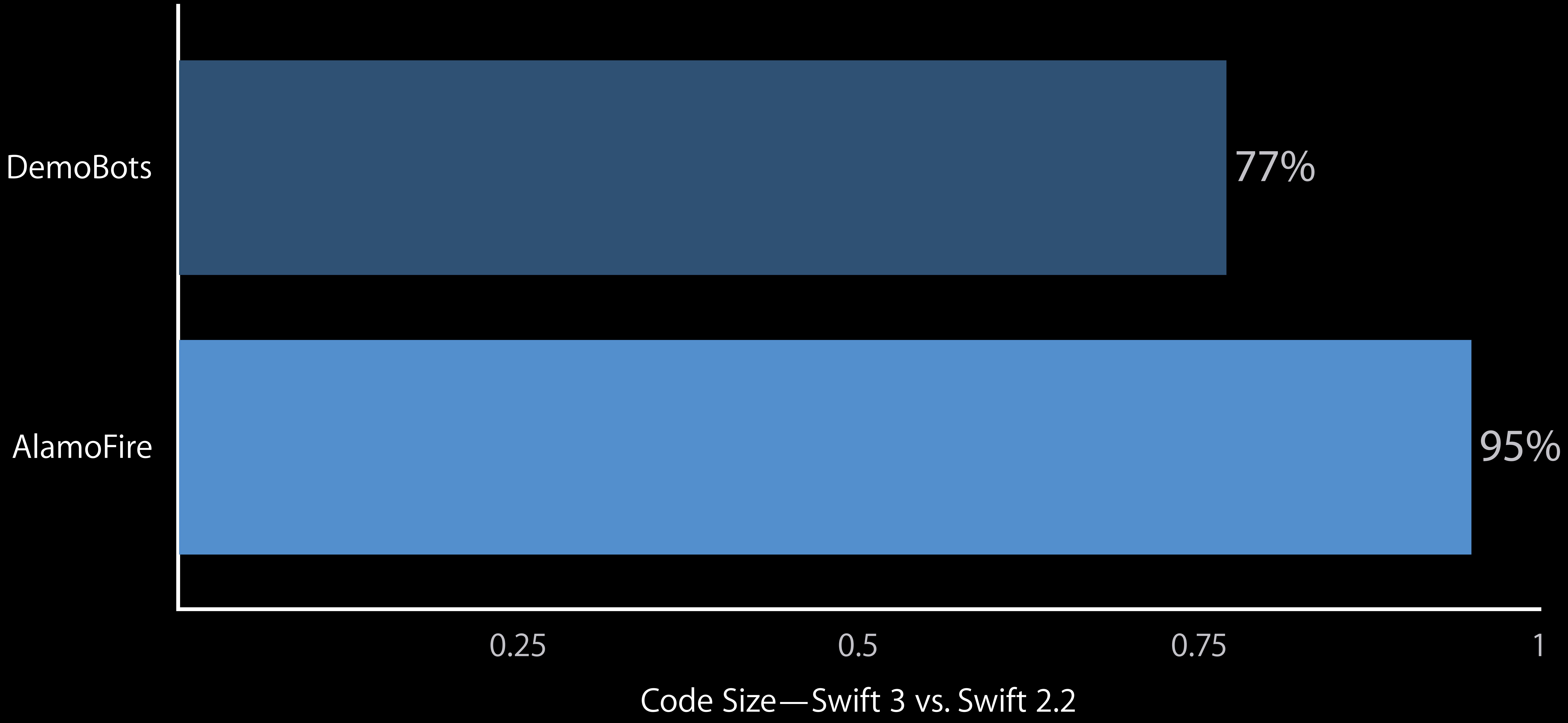
What About Compile Time?



What About Compile Time?

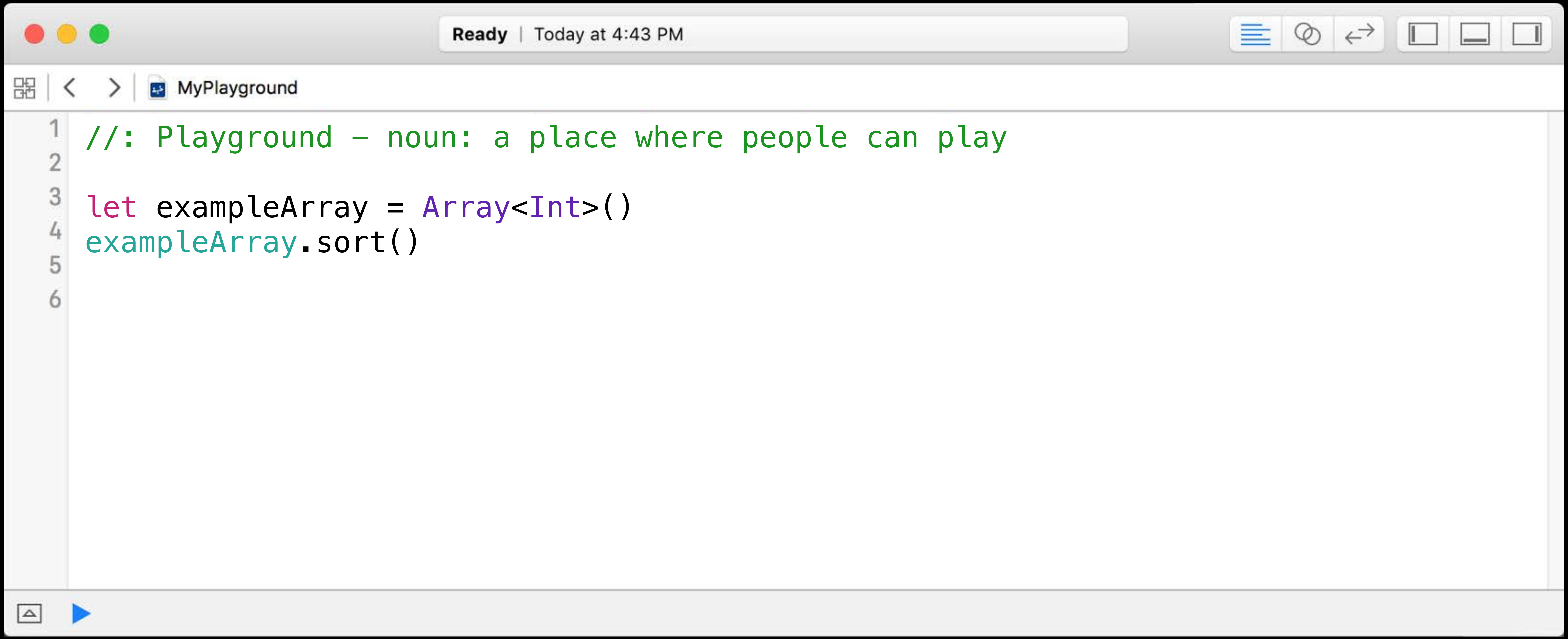


Code Size Optimization



Xcode

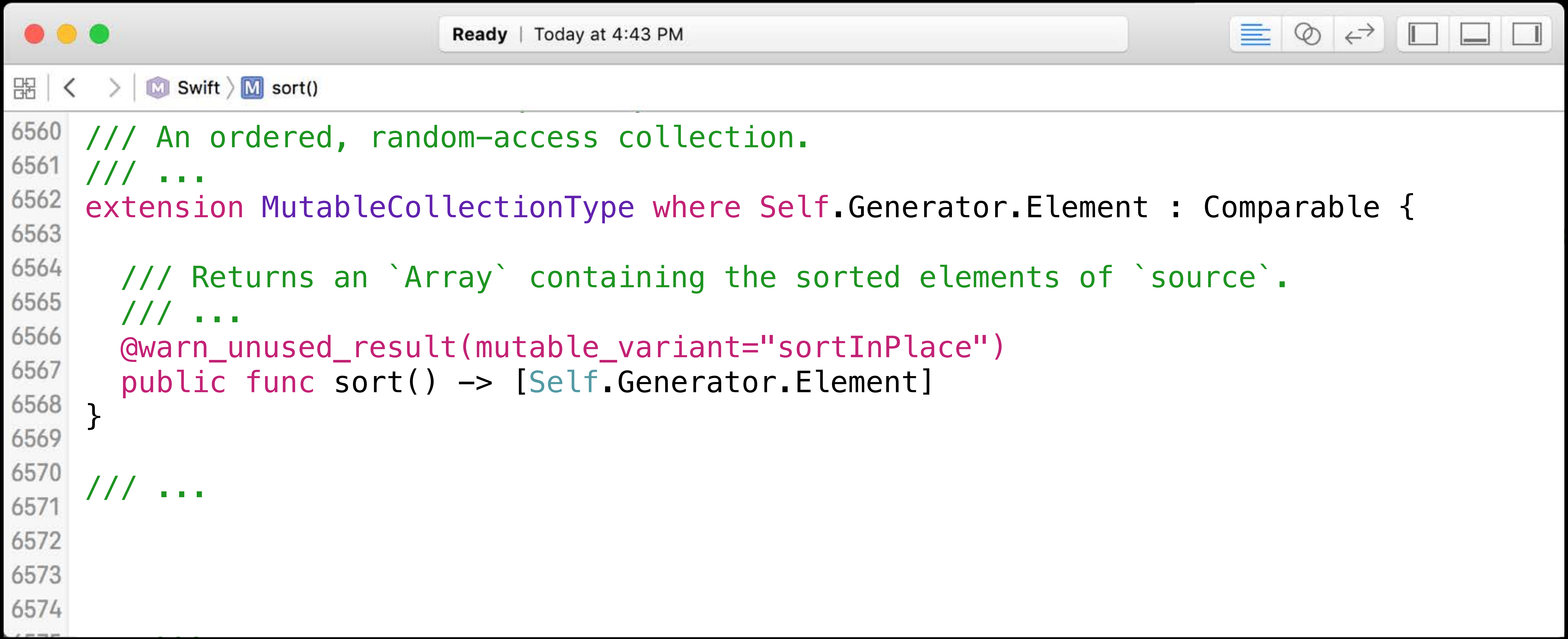
Synthesized Interfaces



```swift
//: Playground - noun: a place where people can play

let exampleArray = Array<Int>()
exampleArray.sort()
```

Synthesized Interfaces

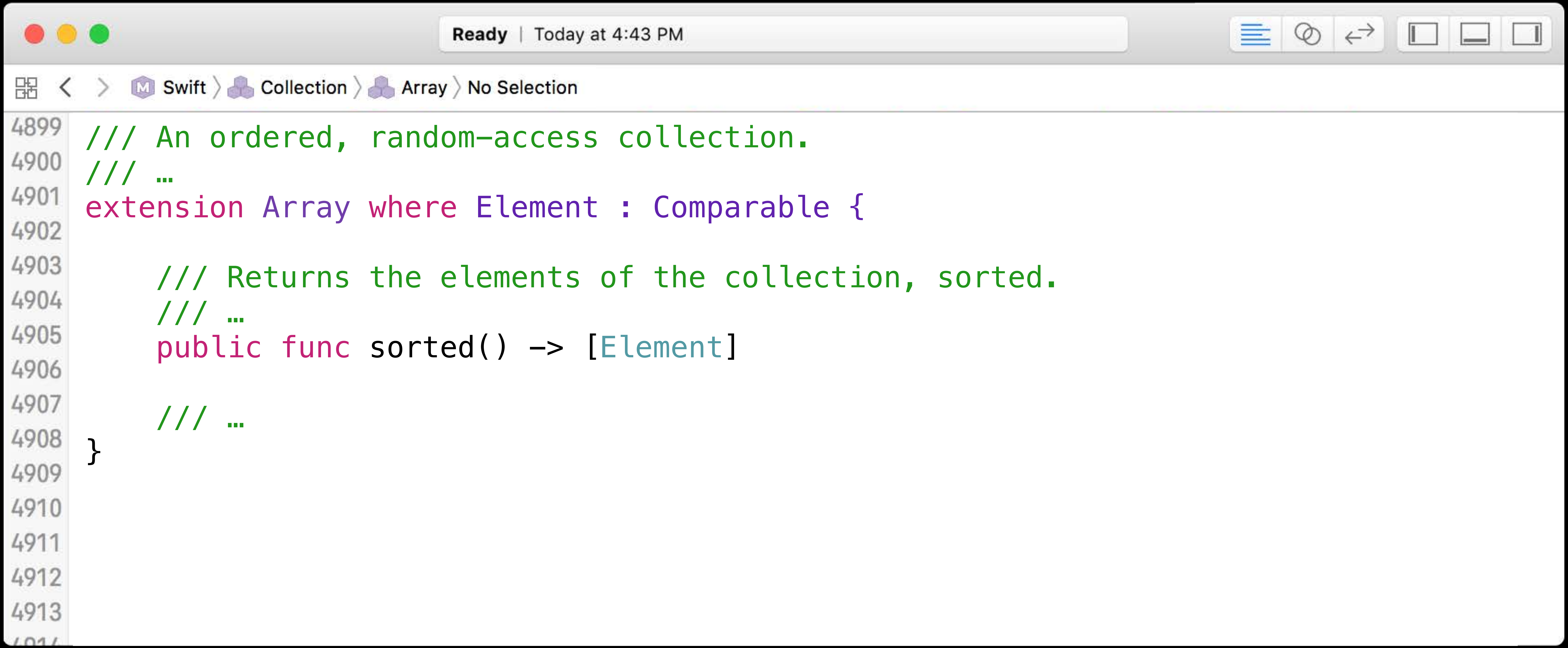


```
/// An ordered, random-access collection.
/// ...
extension MutableCollectionType where Self.Generator.Element : Comparable {

  /// Returns an `Array` containing the sorted elements of `source`.
  /// ...
  @warn_unused_result(mutable_variant="sortInPlace")
  public func sort() -> [Self.Generator.Element]
}

/// ...
```

Flattening Protocols into APIs

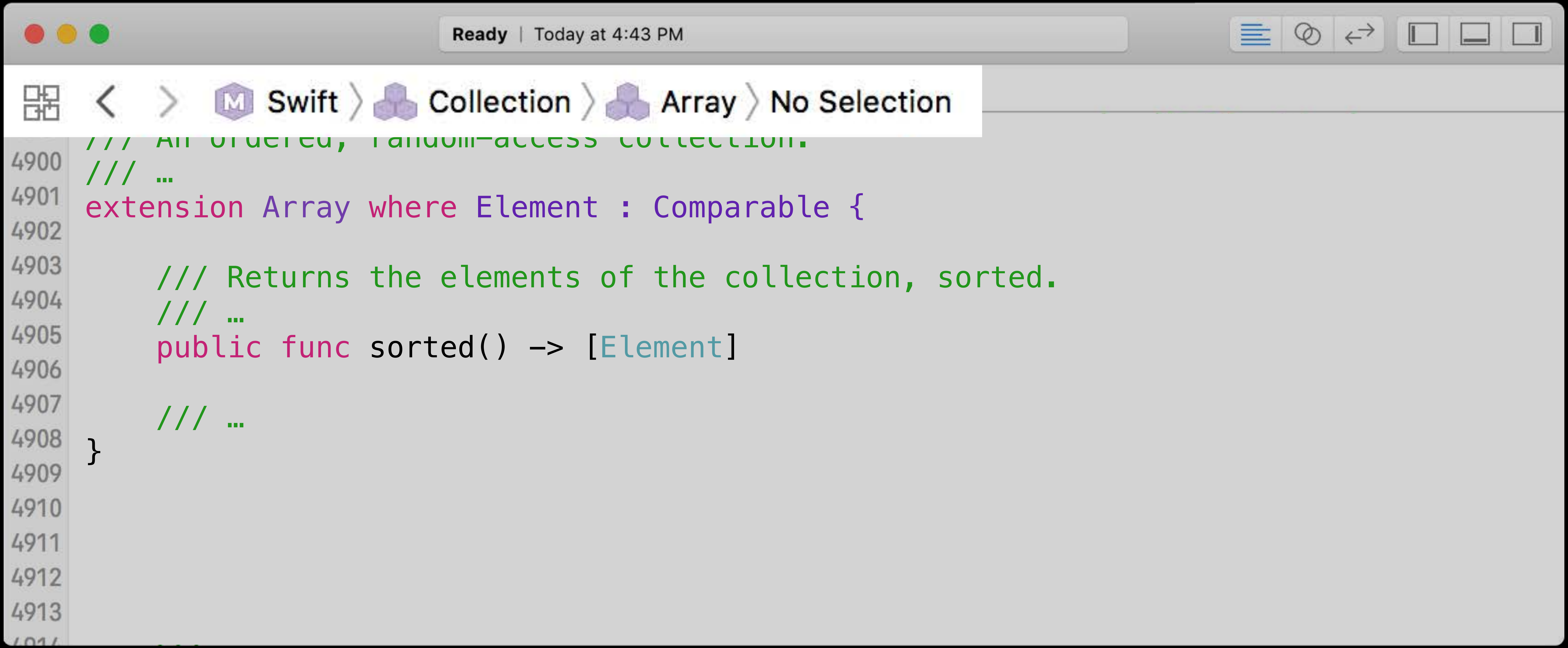


```swift
/// An ordered, random-access collection.
/// …
extension Array where Element : Comparable {

    /// Returns the elements of the collection, sorted.
    /// …
    public func sorted() -> [Element]

    /// …
}
```

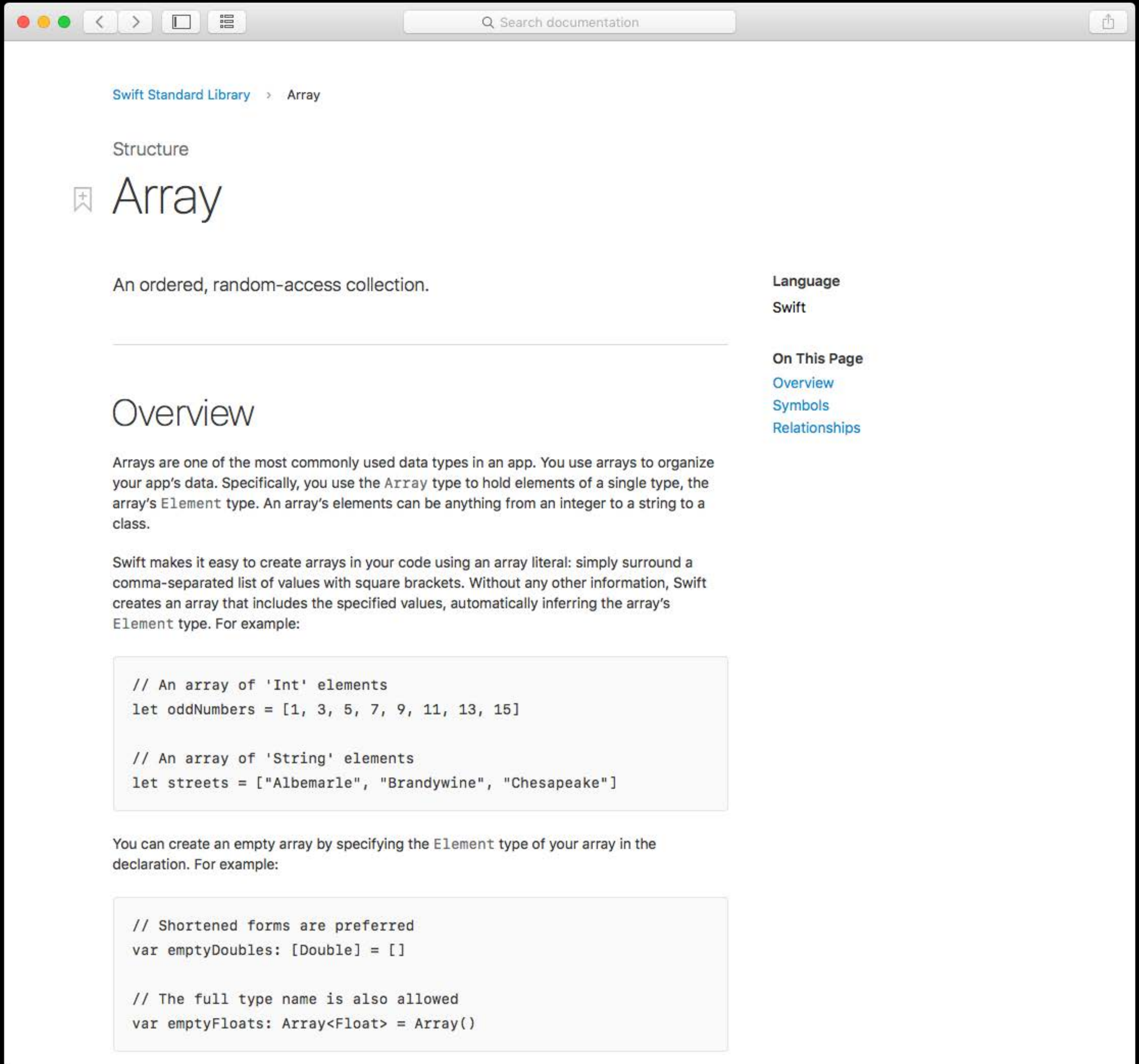
Grouping by Logical Area



```
/// An ordered, random-access collection.
/// …
extension Array where Element : Comparable {

    /// Returns the elements of the collection, sorted.
    /// …
    public func sorted() -> [Element]

    /// …
}
```



Swift Standard Library › Array

Structure

Array

An ordered, random-access collection.

Language

Swift

On This Page

Overview
Symbols
Relationships

Overview

Arrays are one of the most commonly used data types in an app. You use arrays to organize your app's data. Specifically, you use the `Array` type to hold elements of a single type, the array's `Element` type. An array's elements can be anything from an integer to a string to a class.

Swift makes it easy to create arrays in your code using an array literal: simply surround a comma-separated list of values with square brackets. Without any other information, Swift creates an array that includes the specified values, automatically inferring the array's `Element` type. For example:

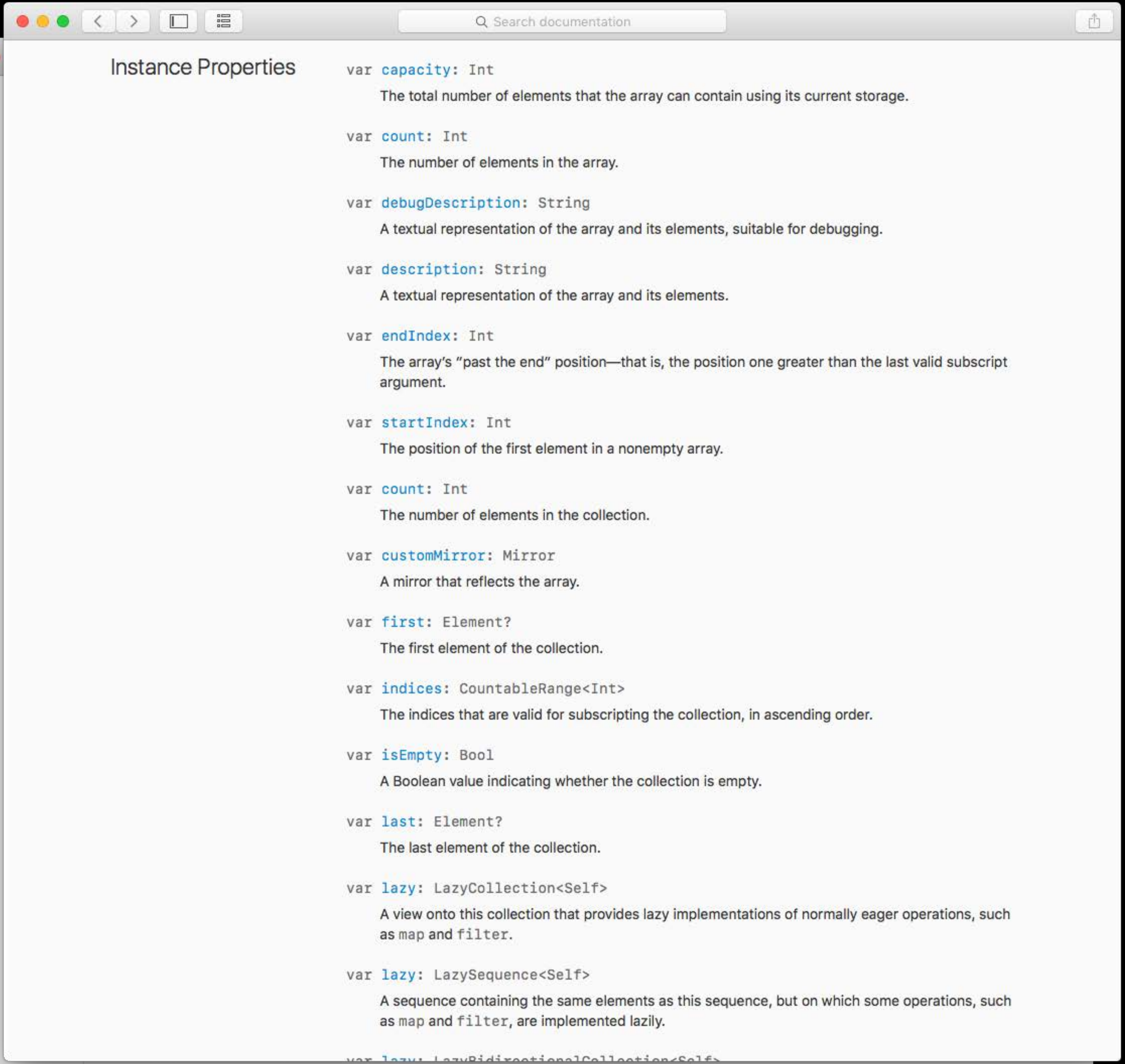
```
// An array of 'Int' elements
let oddNumbers = [1, 3, 5, 7, 9, 11, 13, 15]

// An array of 'String' elements
let streets = ["Albemarle", "Brandywine", "Chesapeake"]
```

You can create an empty array by specifying the `Element` type of your array in the declaration. For example:

```
// Shortened forms are preferred
var emptyDoubles: [Double] = []

// The full type name is also allowed
var emptyFloats: Array<Float> = Array()
```



Instance Properties

`var capacity: Int`
The total number of elements that the array can contain using its current storage.

`var count: Int`
The number of elements in the array.

`var debugDescription: String`
A textual representation of the array and its elements, suitable for debugging.

`var description: String`
A textual representation of the array and its elements.

`var endIndex: Int`
The array's "past the end" position—that is, the position one greater than the last valid subscript argument.

`var startIndex: Int`
The position of the first element in a nonempty array.

`var count: Int`
The number of elements in the collection.

`var customMirror: Mirror`
A mirror that reflects the array.

`var first: Element?`
The first element of the collection.

`var indices: CountableRange<Int>`
The indices that are valid for subscripting the collection, in ascending order.

`var isEmpty: Bool`
A Boolean value indicating whether the collection is empty.

`var last: Element?`
The last element of the collection.

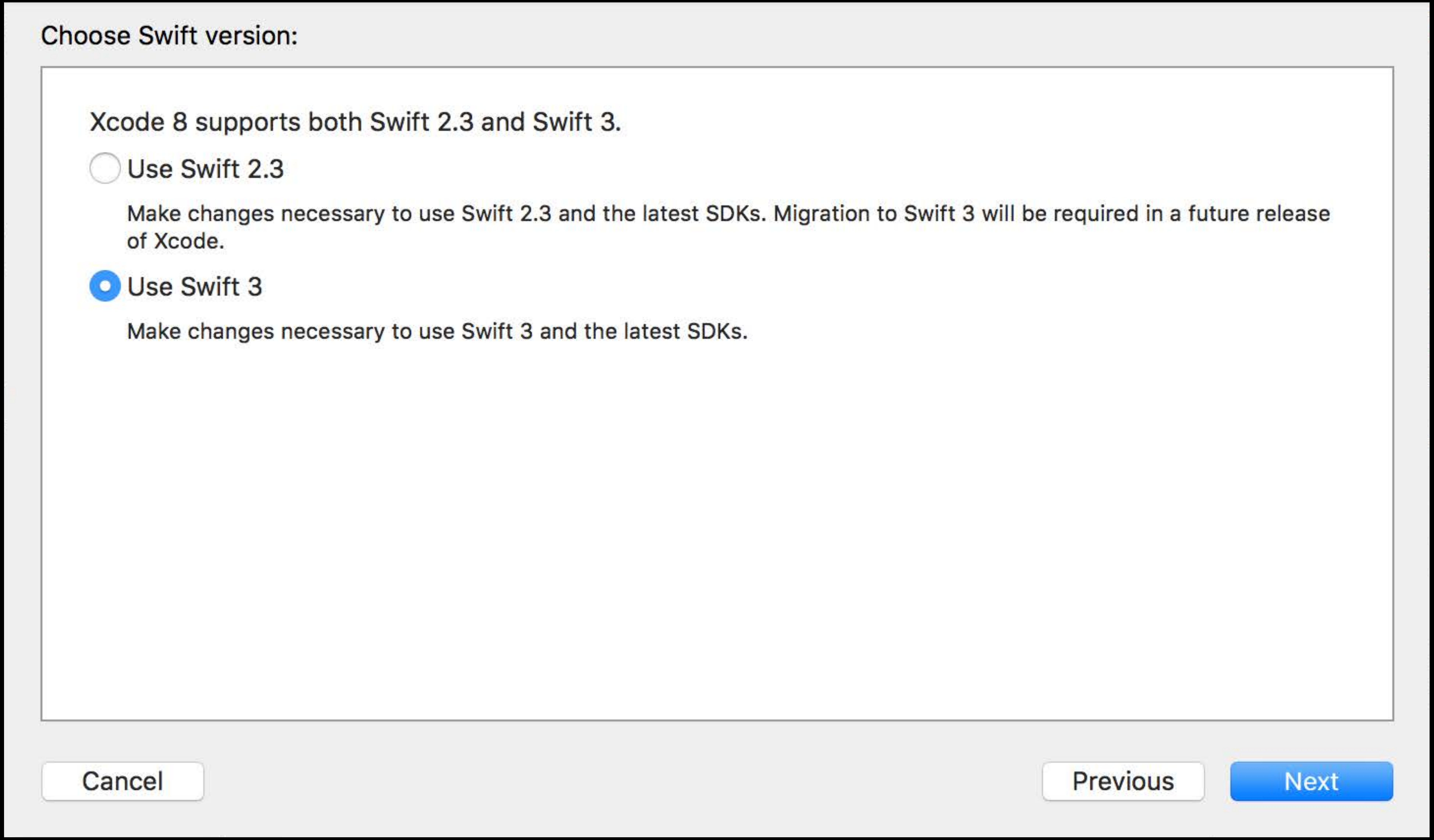
`var lazy: LazyCollection<Self>`
A view onto this collection that provides lazy implementations of normally eager operations, such as `map` and `filter`.

`var lazy: LazySequence<Self>`
A sequence containing the same elements as this sequence, but on which some operations, such as `map` and `filter`, are implemented lazily.

`var lazy: LazyBidirectionalCollection<Self>`

Migrating from Swift 2.2

Migrating from Swift 2.2



Choose Swift version:

Xcode 8 supports both Swift 2.3 and Swift 3.

( ) Use Swift 2.3

Make changes necessary to use Swift 2.3 and the latest SDKs. Migration to Swift 3 will be required in a future release of Xcode.

[✓ Use Swift 3]

Make changes necessary to use Swift 3 and the latest SDKs.

What Is Swift 2.3?

Swift 2.2 + New SDKs = Swift 2.3

What Is Swift 2.3?

What Is Swift 2.3?

Build, test, and submit to App Store fully supported

What Is Swift 2.3?

Build, test, and submit to App Store fully supported

Playgrounds and documentation depend on Swift 3

What Is Swift 2.3?

Build, test, and submit to App Store fully supported

Playgrounds and documentation depend on Swift 3

Interim solution until you migrate to Swift 3

Working with Swift 2.2 and Swift 2.3

```
var groupBackgroundImage: UIImage {
  UIGraphicsBeginImageContextWithOptions(groupBackgroundImageSize, false, 2.0)
  drawCompleteItemsCountInCurrentContext()

  let frame = UIGraphicsGetImageFromCurrentImageContext()
  UIGraphicsEndImageContext()

  return frame
}
```

SE-0020

Working with Swift 2.2 and Swift 2.3

```
var groupBackgroundImage: UIImage {
  UIGraphicsBeginImageContextWithOptions(groupBackgroundImageSize, false, 2.0)
  drawCompleteItemsCountInCurrentContext()

  let frame = UIGraphicsGetImageFromCurrentImageContext()
  UIGraphicsEndImageContext()

  return frame!
}
```

SE-0020

Working with Swift 2.2 and Swift 2.3

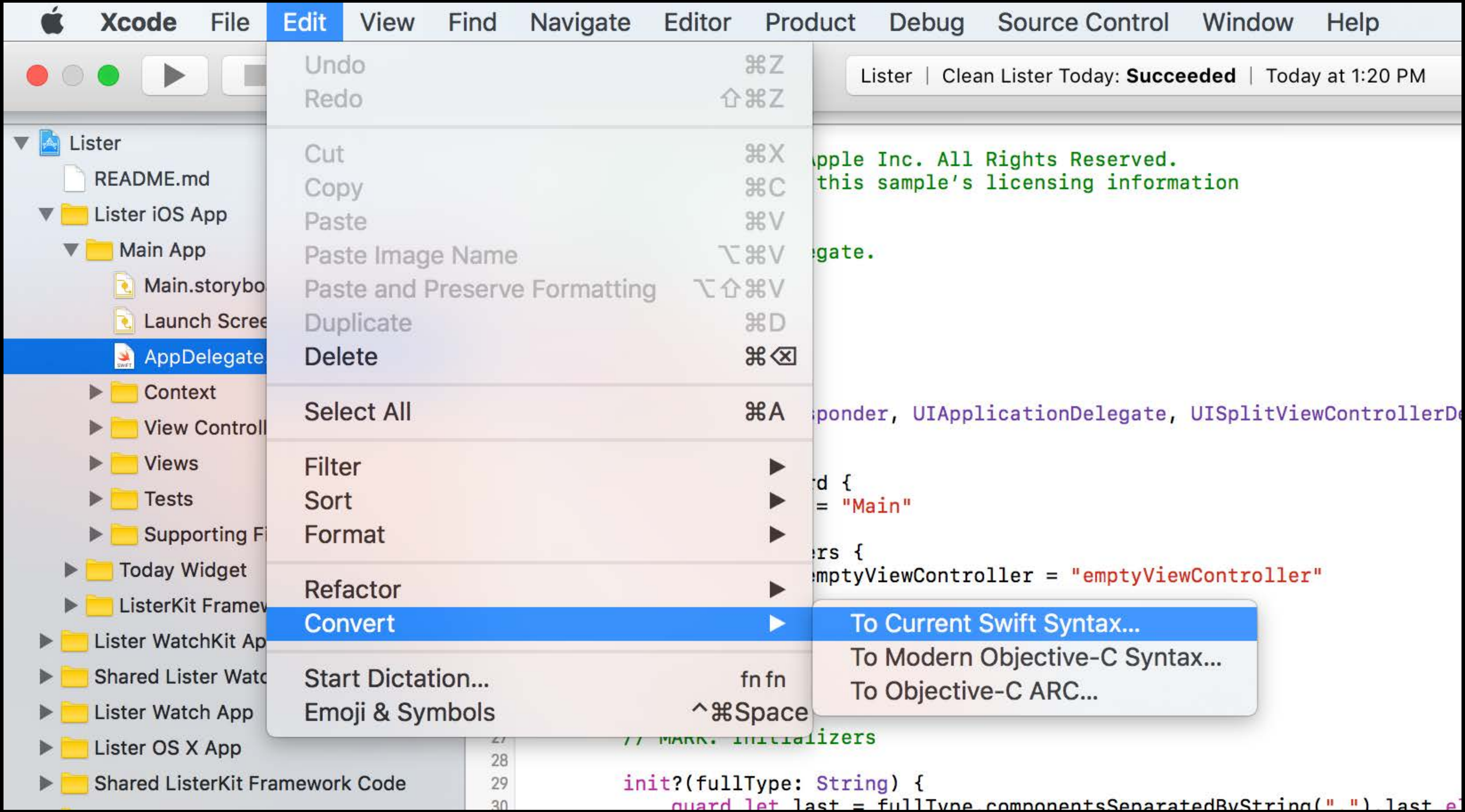
```
var groupBackgroundImage: UIImage {
  UIGraphicsBeginImageContextWithOptions(groupBackgroundImageSize, false, 2.0)
  drawCompleteItemsCountInCurrentContext()

  let frame = UIGraphicsGetImageFromCurrentImageContext()
  UIGraphicsEndImageContext()

  #if swift(>=2.3)
    return frame!
  #else
    return frame
  #endif
}
```

SE-0020

From Swift 2.3 to Swift 3



```
pple Inc. All Rights Reserved.
this sample's licensing information

gate.

ponder, UIApplicationDelegate, UISplitViewControllerDe

d {
= "Main"

rs {
mptyViewController = "emptyViewController"

// MARK: Initializers

init?(fullType: String) {
```

Summary

Swift 3 focuses on fundamentals

See swift.org for how to get involved

Migrator available to Swift 3

More Information

https://developer.apple.com/wwdc16/402

Related Sessions

Swift API Design Guidelines	Presidio	Tuesday 10:00AM
Getting Started with Swift	Pacific Heights	Tuesday 1:40PM
What's New in Foundation for Swift	Mission	Tuesday 4:00PM
Introducing Swift Playgrounds	Mission	Wednesday 11:00AM
Going Server-Side with Swift Open Source	Mission	Friday 9:00AM
Understanding Swift Performance	Mission	Friday 11:00AM
Concurrent Programming with GCD in Swift 3	Pacific Heights	Friday 4:00PM

Labs

Swift Open Hours	Developer Tools Lab A	Tuesday 12:00PM
Swift Open Hours	Developer Tools Lab A	Tuesday 3:00PM
Swift Open Hours	Developer Tools Lab A	Wednesday 9:00AM
Swift Open Hours	Developer Tools Lab A	Wednesday 12:00PM
Swift Open Hours	Developer Tools Lab A	Wednesday 3:00PM
Swift Open Hours	Developer Tools Lab A	Thursday 9:00AM
Swift Open Hours	Developer Tools Lab A	Thursday 12:00PM
Swift Open Hours	Developer Tools Lab A	Thursday 3:00PM

Labs

Swift Open Hours	Developer Tools Lab A	Wednesday 12:00PM
Swift Open Hours	Developer Tools Lab A	Wednesday 3:00PM
Swift Open Hours	Developer Tools Lab A	Thursday 9:00AM
Swift Open Hours	Developer Tools Lab A	Thursday 12:00PM
Swift Open Hours	Developer Tools Lab A	Thursday 3:00PM
Swift Open Hours	Developer Tools Lab A	Friday 9:00AM
Swift Open Hours	Developer Tools Lab A	Friday 12:00PM
Swift Open Hours	Developer Tools Lab A	Friday 3:00PM

WWDC16