#WWDC19

What's New in Swift

Ted Kremenek, Swift Team
Anna Zaks, Swift Team

Released in March

Swift 5

Xcode 10.2

Developer Preview

Swift 5.1

Xcode 11

APIs

Shared Swift Runtime for Apps

Binary Frameworks

Apple Swift-Only Frameworks

Package Manager in Xcode

More API Expressibility

Library Evolution

APIs

Shared Swift Runtime for Apps

Binary Frameworks

Apple Swift-Only Frameworks

Package Manager in Xcode

More API Expressibility

Library Evolution

APIs

ABI and Module Stability

What is ABI?

ABI = “Application Binary Interface”

Specifies details of a program’s representation at runtime

Compatible ABIs allows separately compiled code to interact at runtime


exec


exec


exec
Running process

Before ABI stability these both need to be built with the same compiler!

ABI stability in Swift 5

Program and framework can be built with different compilers (Swift 5 or later)

What is Module Stability?

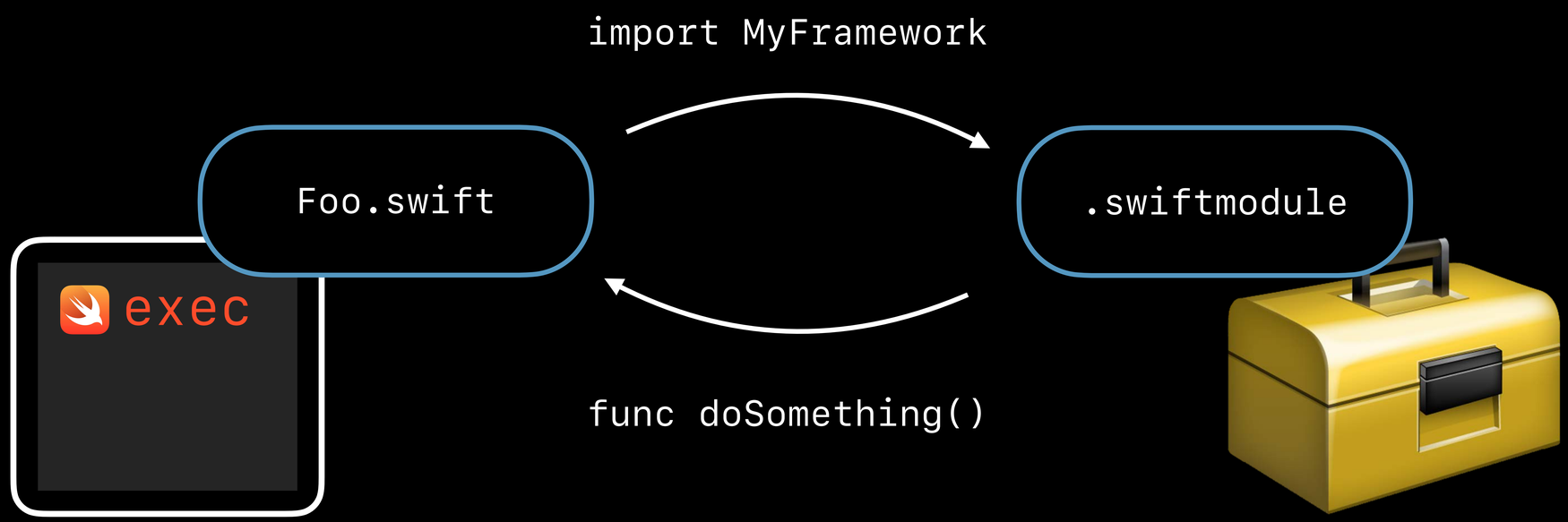
Swift libraries and the APIs they export are known as “modules”

Module files are created (and used) by the compiler

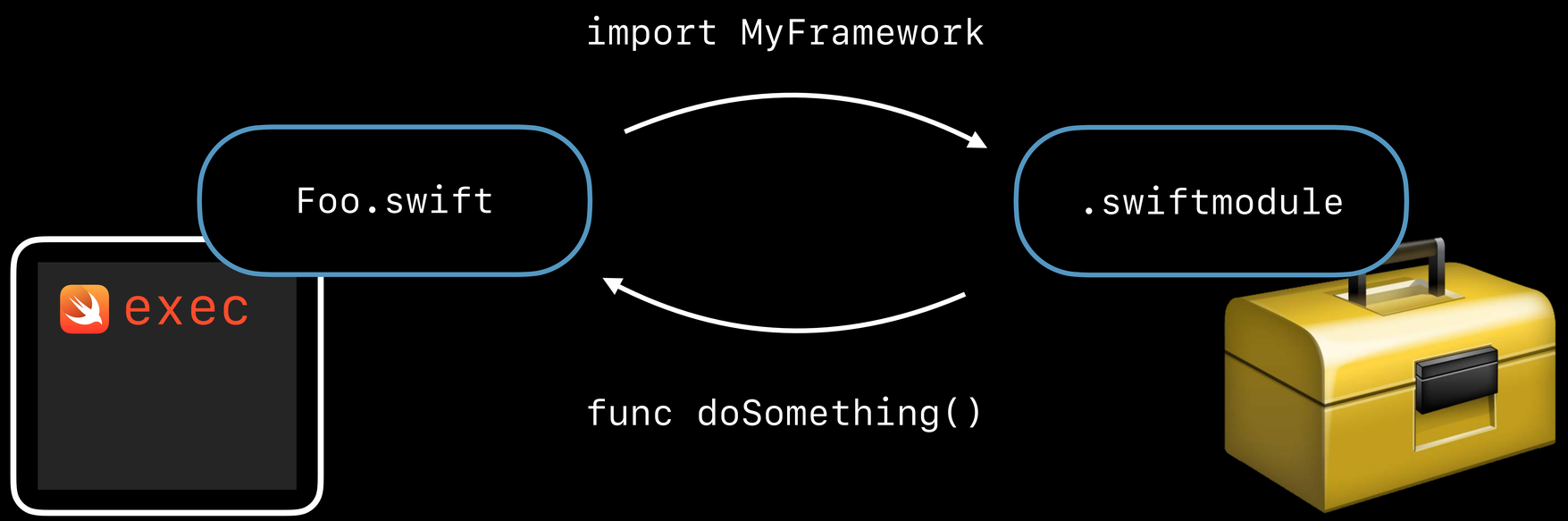

exec


exec

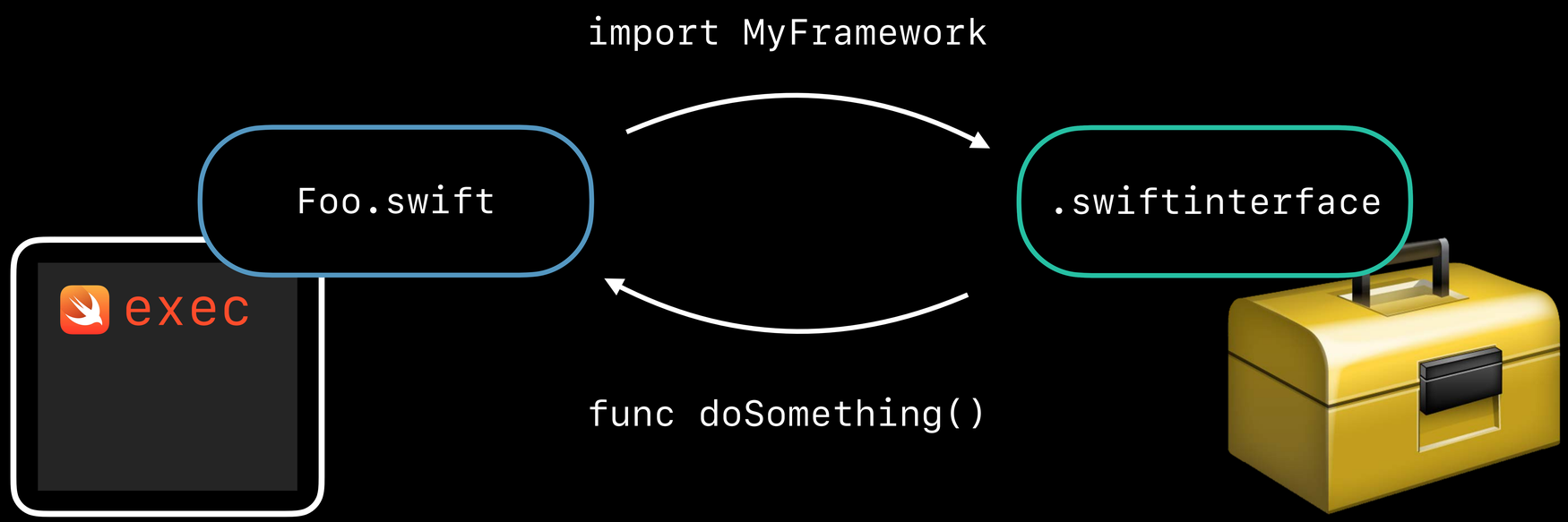
Before module stability these both need to be built with the same compiler!



Swift 5.1 introduces a stable and textual module interface file



Swift 5.1 introduces a stable and textual module interface file



Module + ABI stability
= Binary frameworks

swift.org

Swift

ABOUT SWIFT
BLOG
DOWNLOAD
GETTING STARTED
DOCUMENTATION
SOURCE CODE
COMMUNITY
CONTRIBUTING
CONTINUOUS INTEGRATION
SOURCE COMPATIBILITY

FOCUS AREAS

GSOC 2019
SERVER WORK GROUP

PROJECTS

COMPILER AND STANDARD LIBRARY

ABI Stability and More

FEBRUARY 7, 2019  Jordan Rose

It has been a longstanding goal to stabilize Swift's ABI on macOS, iOS, watchOS, and tvOS. While a stable ABI is an important milestone for the maturity of any language, the ultimate benefit to the Swift ecosystem was to enable binary compatibility for apps and libraries. This post describes what binary compatibility means in Swift 5 and how it will evolve in future releases of Swift.

You may ask: what about other platforms? ABI stability is implemented for each operating system that it compiles and runs on. Swift's ABI is currently declared stable for Swift 5 on Apple platforms. As development of Swift on Linux, Windows, and other platforms matures, the Swift Core Team will evaluate stabilizing the ABI on those platforms.

Swift 5 provides binary compatibility for apps: a guarantee that going forward, an app built with one version of the Swift compiler will be able to talk to a library built with another version. This applies even when using the compatibility mode with older language versions (`-swift-version 4.2`).

swift.org

Swift

- ABOUT SWIFT
- BLOG
- DOWNLOAD
- GETTING STARTED
- DOCUMENTATION
- SOURCE CODE
- COMMUNITY
- CONTRIBUTING
- CONTINUOUS INTEGRATION
- SOURCE COMPATIBILITY

FOCUS AREAS

- GSOC 2019
- SERVER WORK GROUP

PROJECTS

- COMPILER AND STANDARD LIBRARY

ABI Stability and More

FEBRUARY 7, 2019 Jordan Rose

It has been a longstanding goal to stabilize Swift's ABI on macOS, iOS, watchOS, and tvOS. While a stable ABI is an important milestone for the maturity of any language, the ultimate benefit to the Swift ecosystem was to enable binary compatibility for apps and libraries. This post describes what binary compatibility means in Swift 5 and how it will evolve in future releases of Swift.

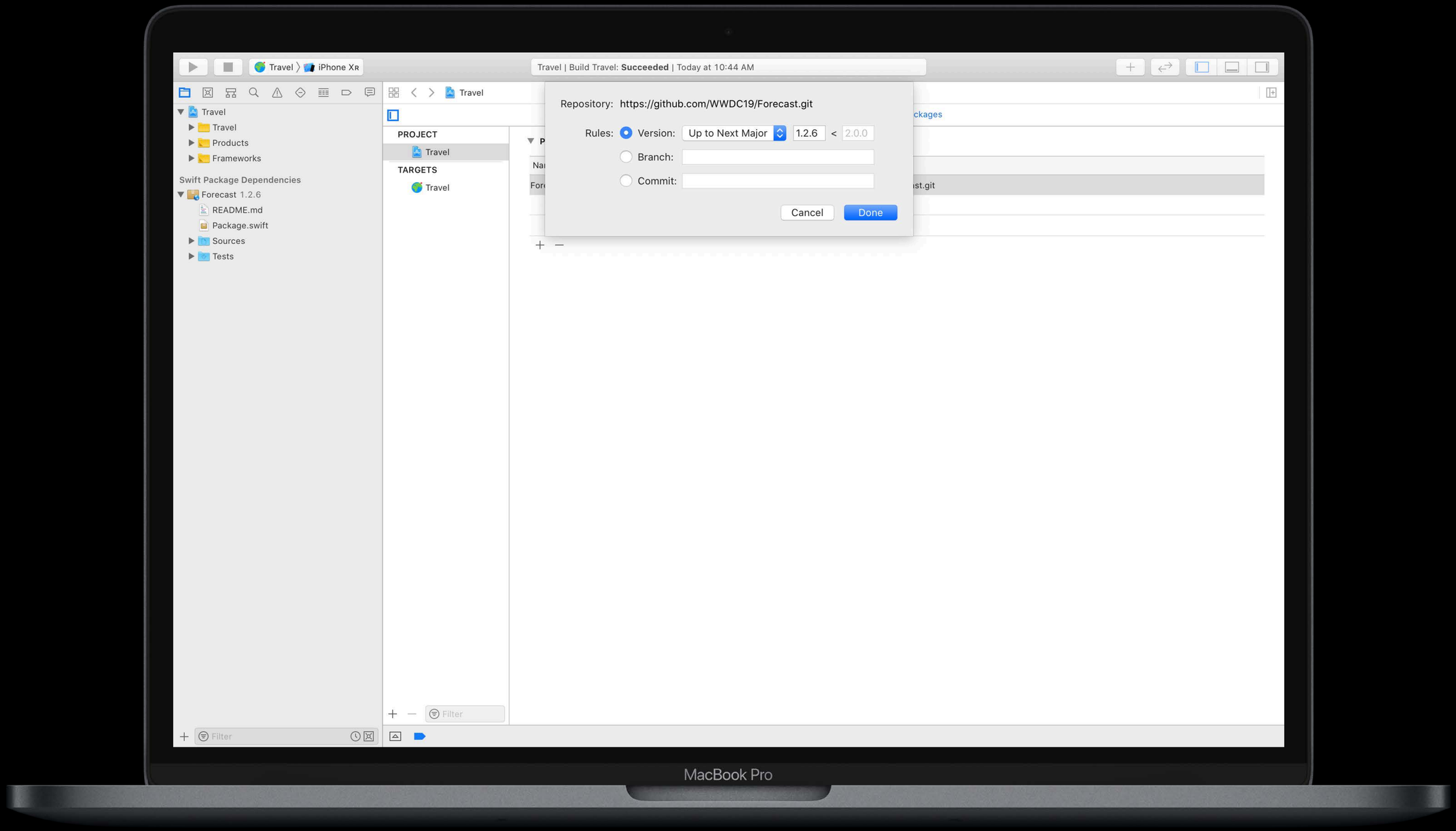
You may ask: what about other platforms? ABI stability is implemented for each operating system [illegible] and [illegible] currently declared stable for Swift 5 on Apple platforms. As development of Swift on Linux, Windows, and other platforms matures, the Swift Core Team will evaluate stabilizing the ABI on those platforms.

Swift 5 provides binary compatibility for apps: a guarantee that going forward, an app built with one version of the Swift compiler will be able to talk to a library built with another version. This applies even when using the compatibility mode with older language versions (`-swift-version 4.2`).

swift.org/blog

Binary Frameworks in Swift

Thursday, 3:00 PM


Travel 〉 iPhone XR
Travel | Build Travel: Succeeded | Today at 10:44 AM
Travel
Travel
Products
Frameworks
Swift Package Dependencies
Forecast 1.2.6
README.md
Package.swift
Sources
Tests
PROJECT
Travel
TARGETS
Travel
Repository: https://github.com/WWDC19/Forecast.git
Rules: Version: Up to Next Major 1.2.6 < 2.0.0
Branch:
Commit:
Cancel
Done
Filter
MacBook Pro

Adopting Swift Packages in Xcode	Wednesday, 3:00 PM
Creating Swift Packages	Thursday, 9:00AM

Performance

Shared Swift Runtime for Apps in the OS

macOS	iOS	tvOS	watchOS
10.14.4	12.2	12.2	5.2

Apps and the Shared Swift Runtime in the OS

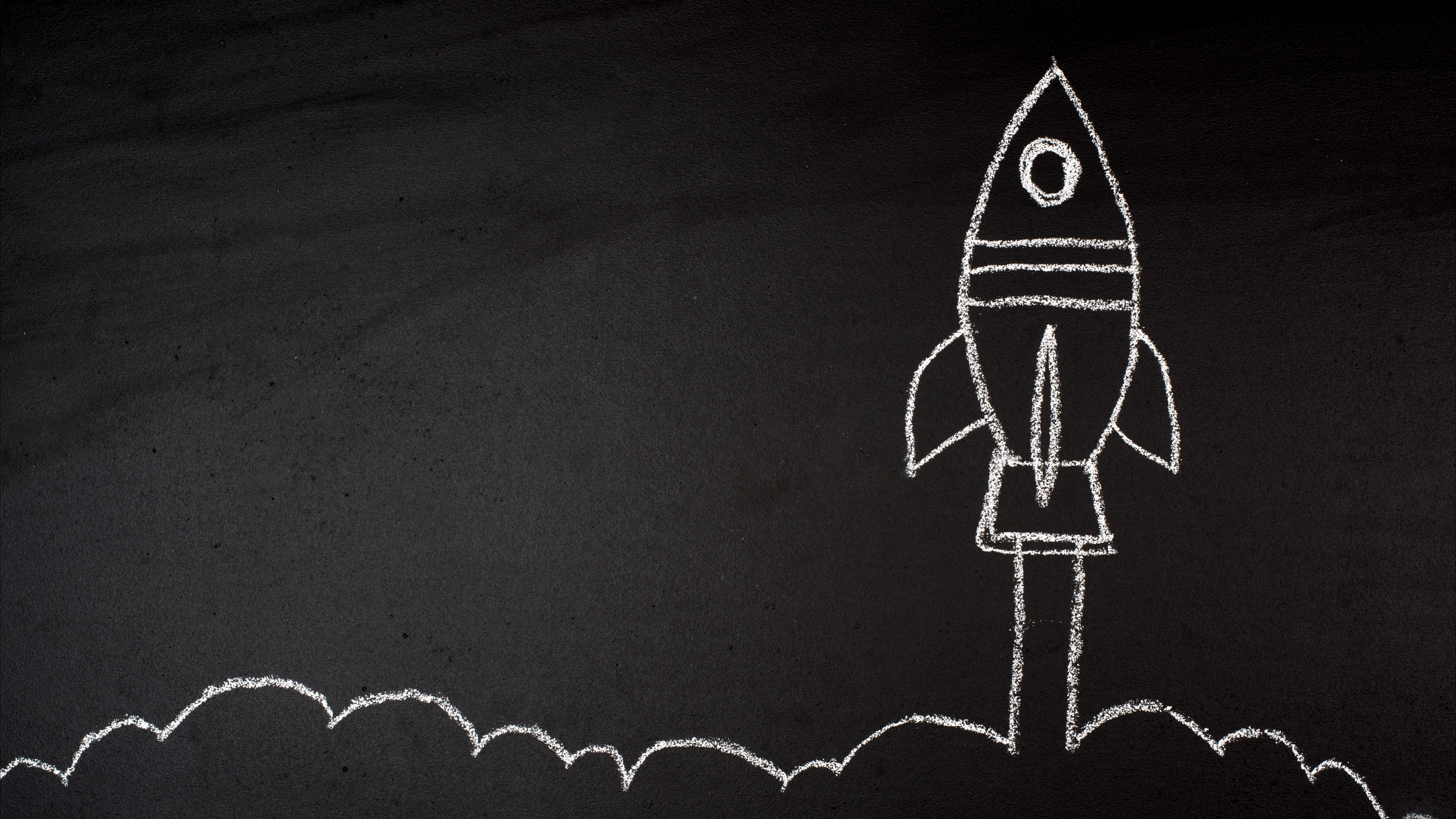
Apps use the runtime from OS when it is available

Apps and the Shared Swift Runtime in the OS

Apps use the runtime from OS when it is available

Apps that backward deploy contain a runtime copy

- Embedded runtime copy is inert on OSs with shared runtime
- iOS App Store thins out runtime copy when downloading to iOS 12.2 or later



5%

Launch time **overhead** with **Swift 4.2**

5%

Launch time **overhead** with **Swift 4.2**

5%

Launch time overhead with **Swift 5**

0%

Launch time overhead with **Swift 5**

Code Size Reductions

10%

Smaller Code Size

15%

Smaller Code Size when using ‘Optimize for Size’

Faster Bridging Between Swift and Objective-C

Swift Bridging of Currency Types Used in Cocoa

Objective-C

```
- (BOOL)unlockWithPassword:(NSString *)password;
```

Swift

```
func unlock(withPassword password: String) -> Bool
```

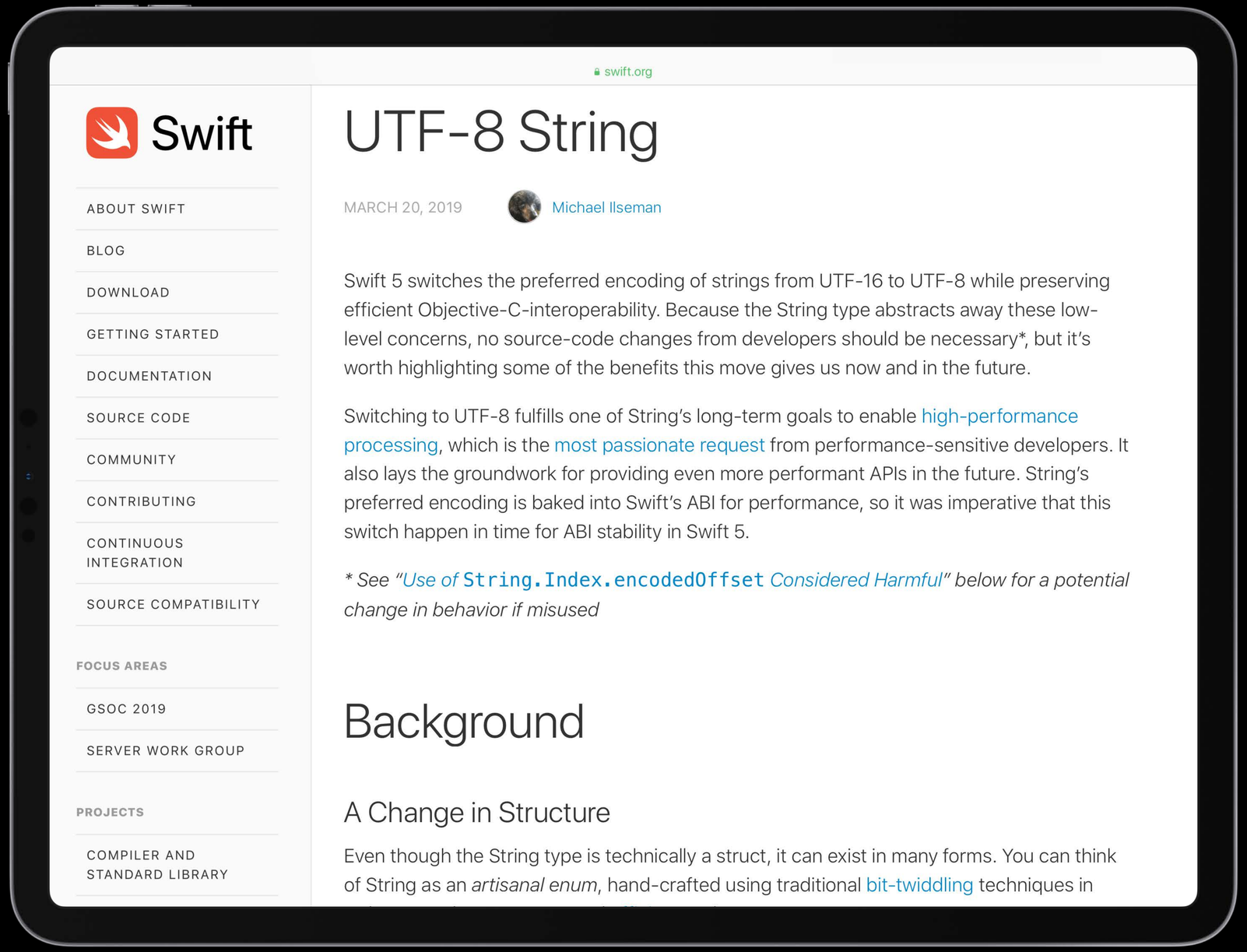
1.6x

NSDictionary to Dictionary Bridging

15x

NSString operations when performed on bridged Swift Strings

`String`


swift.org
Swift
ABOUT SWIFT
BLOG
DOWNLOAD
GETTING STARTED
DOCUMENTATION
SOURCE CODE
COMMUNITY
CONTRIBUTING
CONTINUOUS INTEGRATION
SOURCE COMPATIBILITY
FOCUS AREAS
GSOC 2019
SERVER WORK GROUP
PROJECTS
COMPILER AND STANDARD LIBRARY
UTF-8 String
MARCH 20, 2019
Michael Ilseman
Swift 5 switches the preferred encoding of strings from UTF-16 to UTF-8 while preserving efficient Objective-C-interoperability. Because the String type abstracts away these low-level concerns, no source-code changes from developers should be necessary*, but it's worth highlighting some of the benefits this move gives us now and in the future.
Switching to UTF-8 fulfills one of String's long-term goals to enable high-performance processing, which is the most passionate request from performance-sensitive developers. It also lays the groundwork for providing even more performant APIs in the future. String's preferred encoding is baked into Swift's ABI for performance, so it was imperative that this switch happen in time for ABI stability in Swift 5.
* See "Use of String.Index.encodedOffset Considered Harmful" below for a potential change in behavior if misused
Background
A Change in Structure
Even though the String type is technically a struct, it can exist in many forms. You can think of String as an artisanal enum, hand-crafted using traditional bit-twiddling techniques in

Native Swift Strings now interoperate with C APIs without overhead

Native Swift Strings now interoperate with C APIs without overhead

电池续航

Small Strings now include Unicode characters in addition to ASCII

Native Swift Strings now interoperate with C APIs without overhead

电池续航

Small Strings now include Unicode characters in addition to ASCII

`NSString` ⇆ `String`

Fast Interoperability with NSString

SwiftNIO

SwiftNIO is a cross-platform asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients.

It's like Netty, but written for Swift.

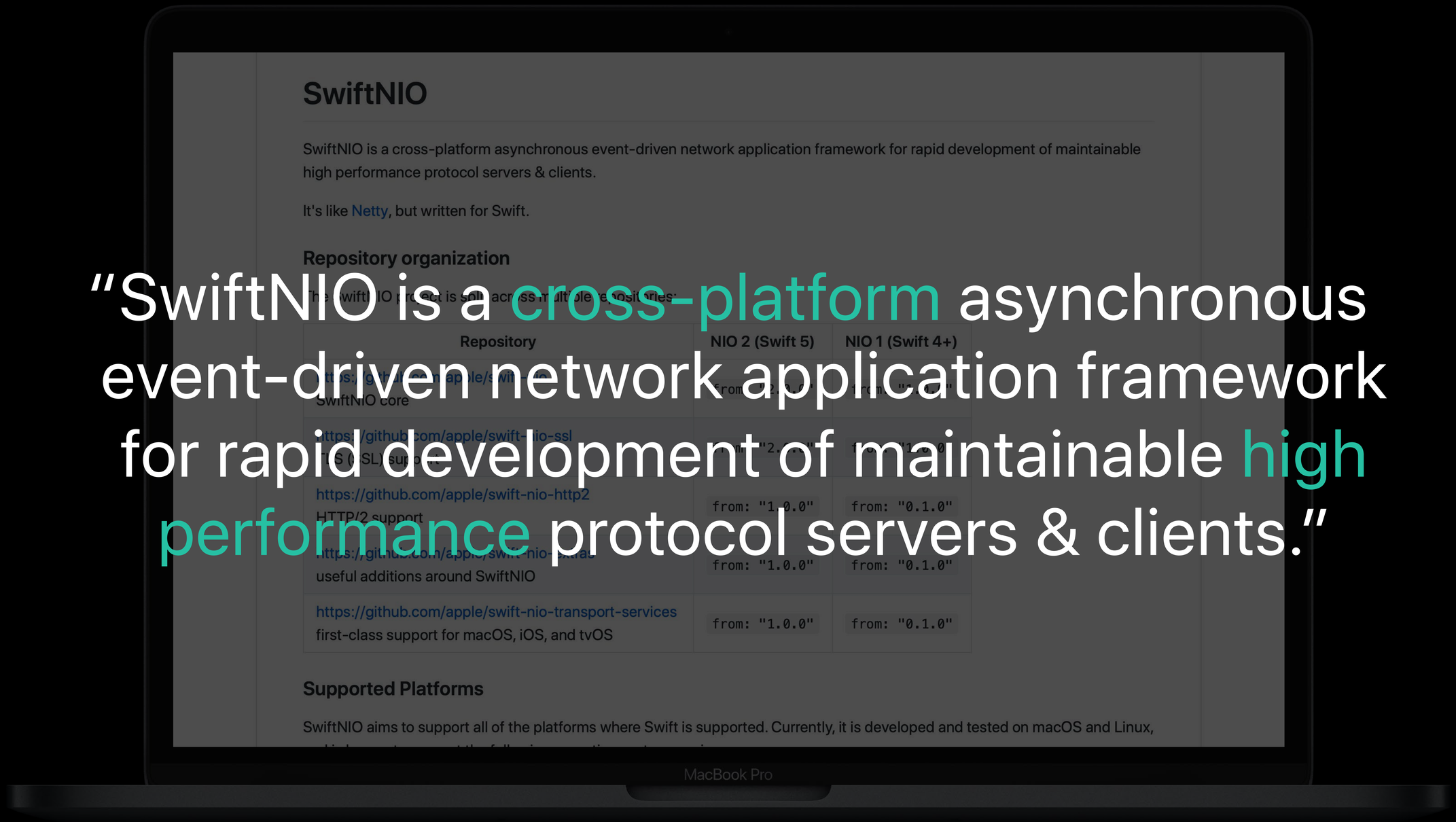
Repository organization

The SwiftNIO project is split across multiple repositories:

Repository	NIO 2 (Swift 5)	NIO 1 (Swift 4+)
https://github.com/apple/swift-nio SwiftNIO core	`from: "2.0.0"`	`from: "1.0.0"`
https://github.com/apple/swift-nio-ssl TLS (SSL) support	`from: "2.0.0"`	`from: "1.0.0"`
https://github.com/apple/swift-nio-http2 HTTP/2 support	`from: "1.0.0"`	`from: "0.1.0"`
https://github.com/apple/swift-nio-extras useful additions around SwiftNIO	`from: "1.0.0"`	`from: "0.1.0"`
https://github.com/apple/swift-nio-transport-services first-class support for macOS, iOS, and tvOS	`from: "1.0.0"`	`from: "0.1.0"`

Supported Platforms

SwiftNIO aims to support all of the platforms where Swift is supported. Currently, it is developed and tested on macOS and Linux,

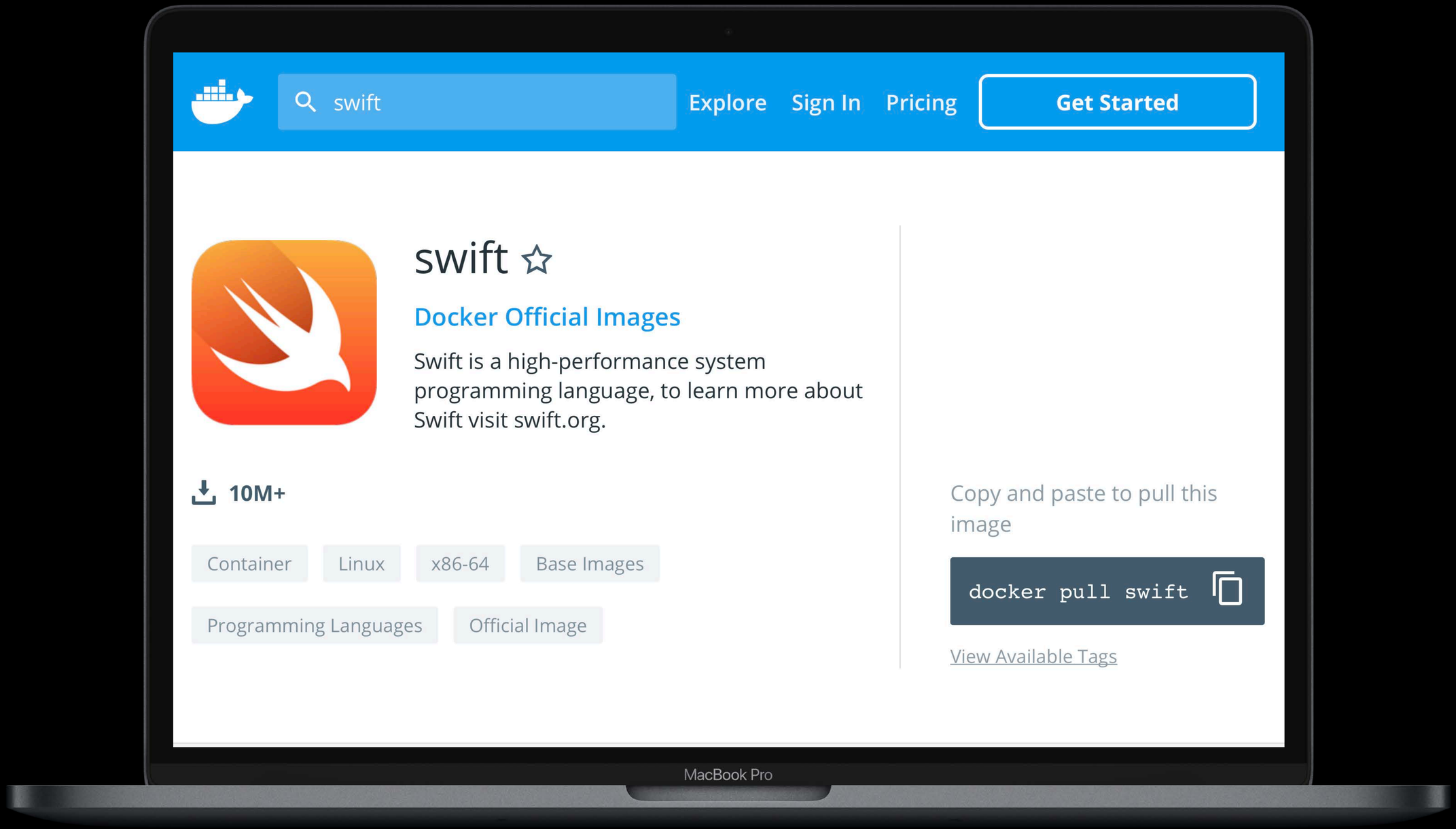


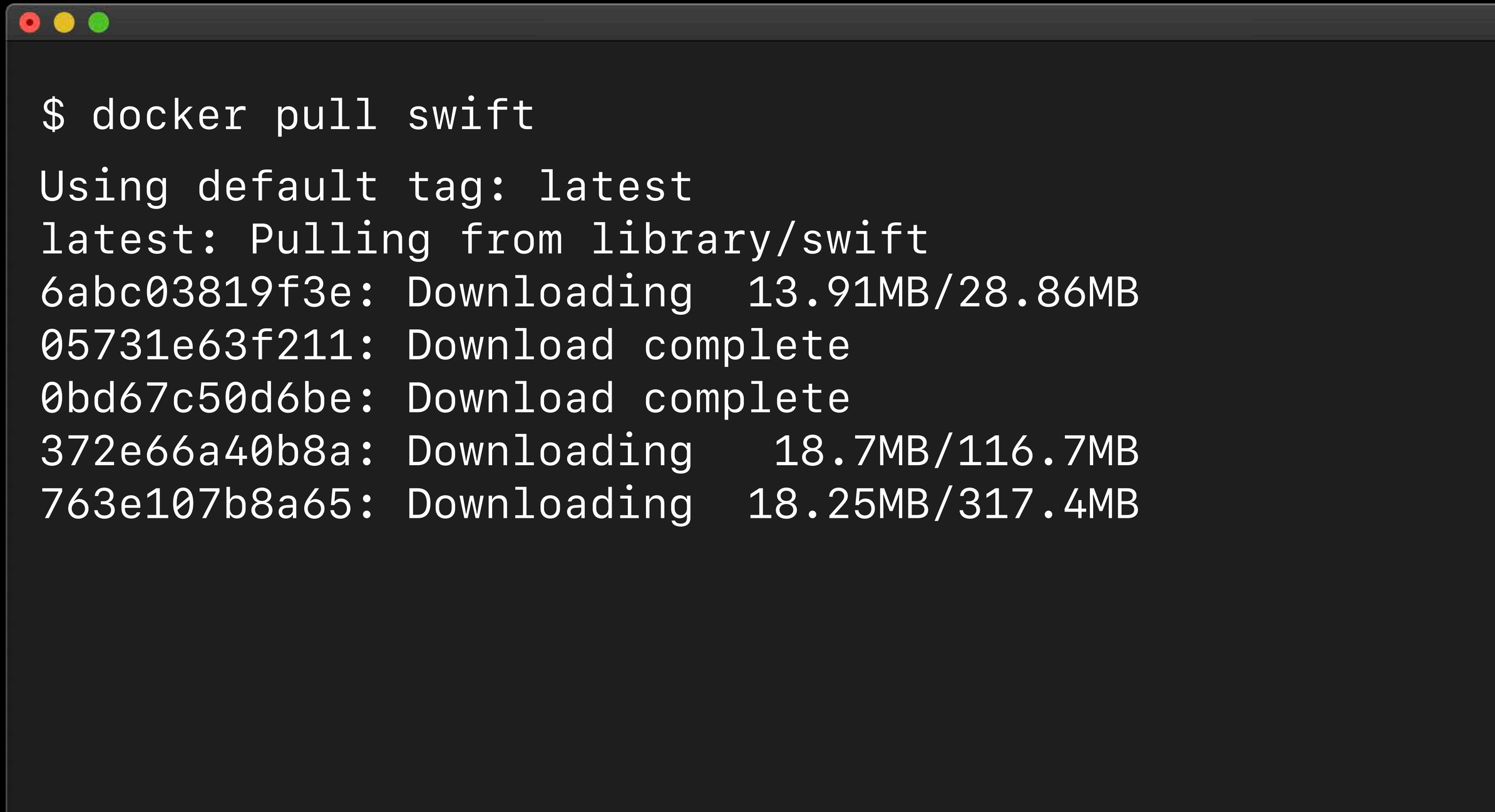
"SwiftNIO is a cross-platform asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients."

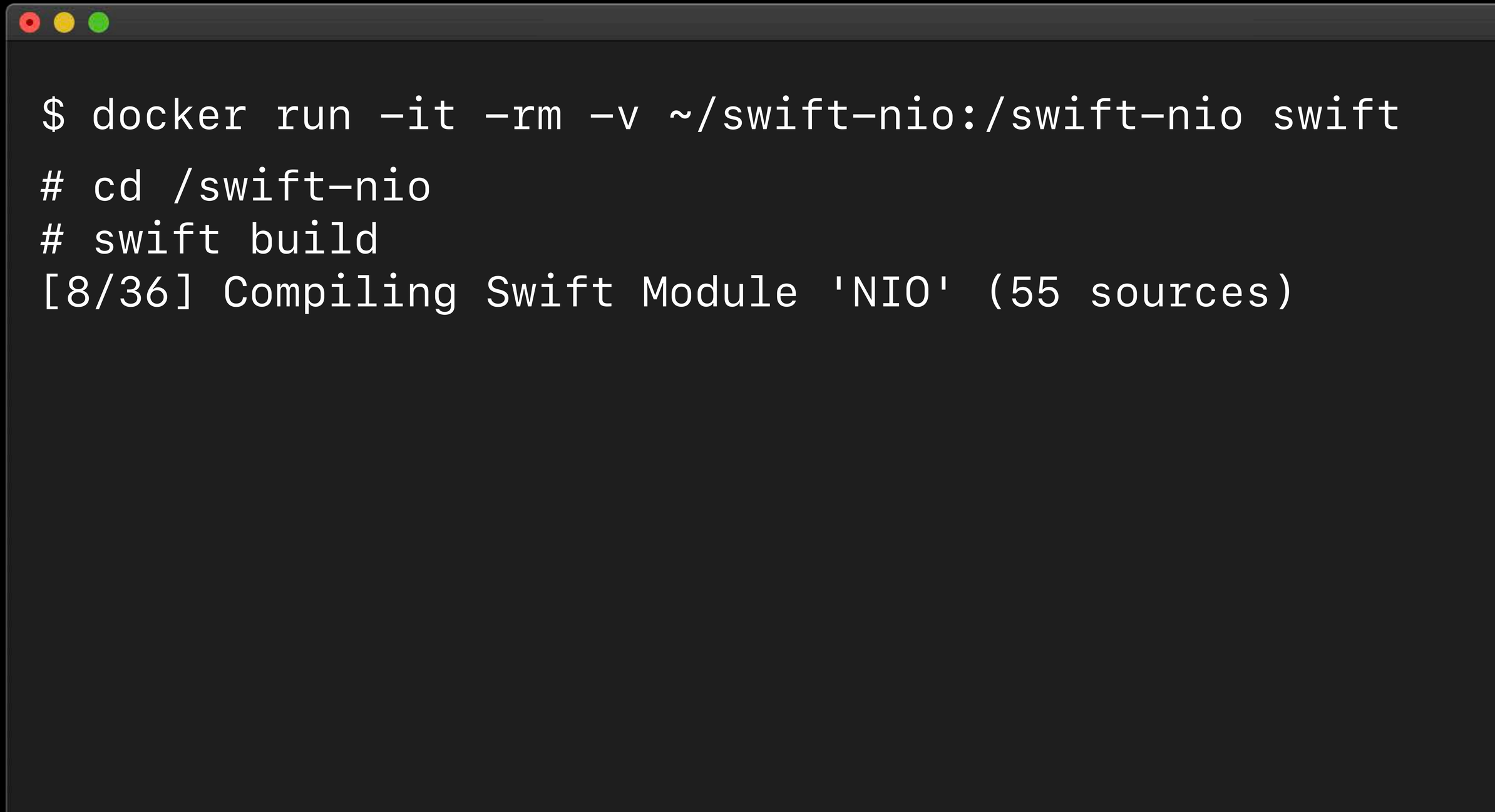
20%

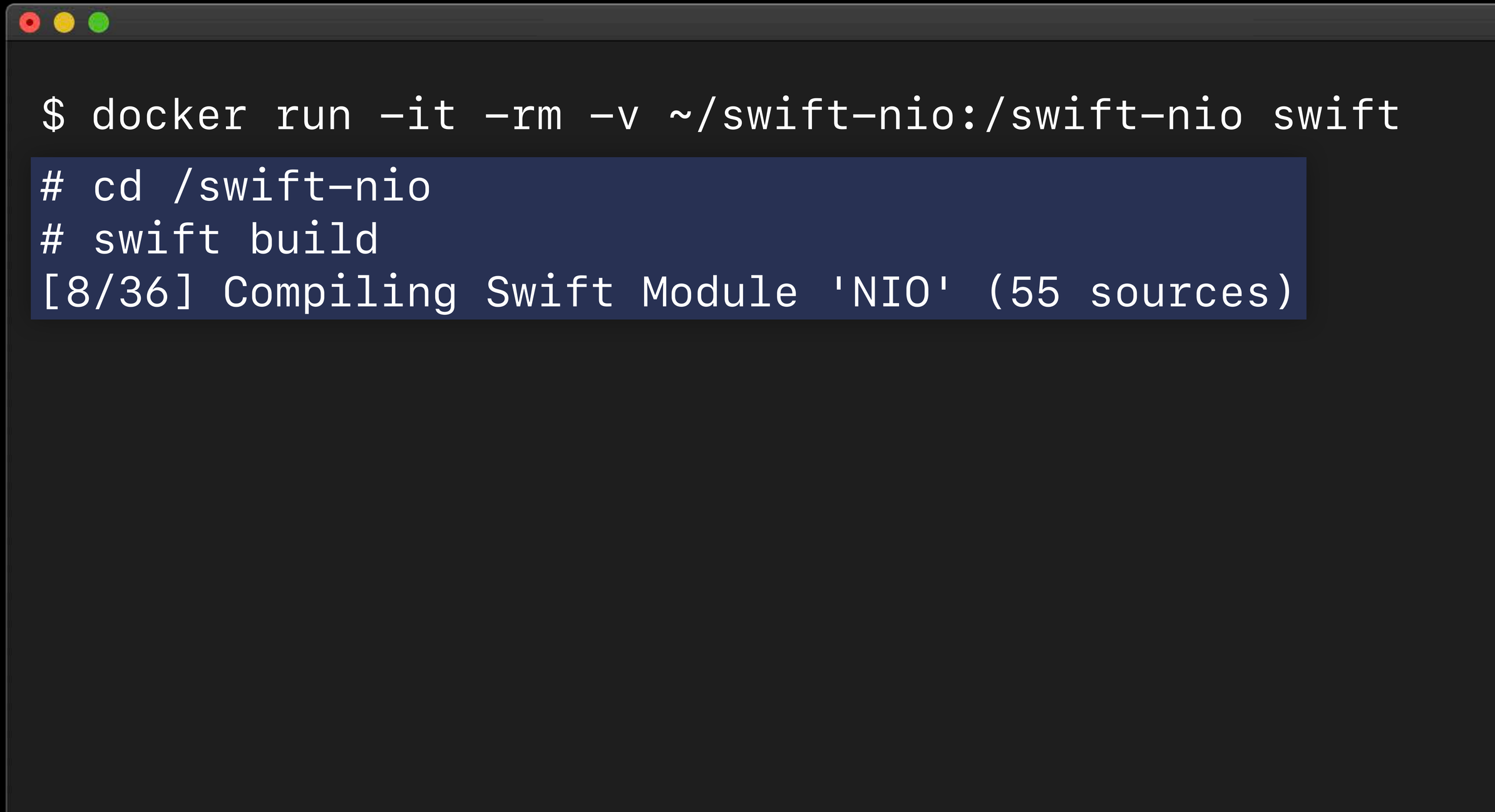
More requests per second on HTTP/1.1 server built on SwiftNIO

Swift Tooling and Open Source


swift
Explore
Sign In
Pricing
Get Started
swift ☆
Docker Official Images
Swift is a high-performance system programming language, to learn more about Swift visit swift.org.
10M+
Container
Linux
x86-64
Base Images
Programming Languages
Official Image
Copy and paste to pull this image
docker pull swift
View Available Tags
MacBook Pro


$ docker pull swift
Using default tag: latest
latest: Pulling from library/swift
6abc03819f3e: Downloading 13.91MB/28.86MB
05731e63f211: Download complete
0bd67c50d6be: Download complete
372e66a40b8a: Downloading 18.7MB/116.7MB
763e107b8a65: Downloading 18.25MB/317.4MB

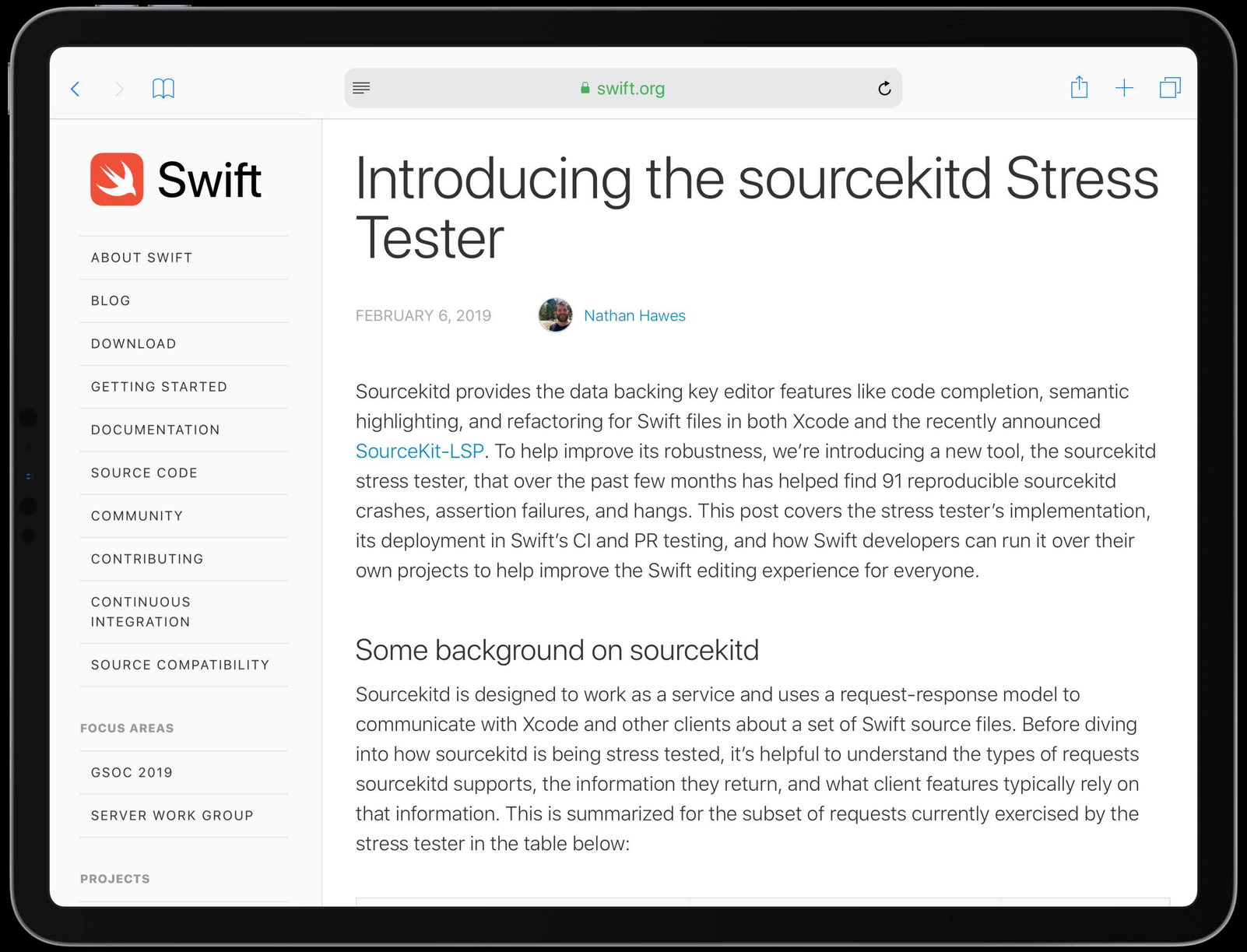

$ docker run -it -rm -v ~/swift-nio:/swift-nio swift
# cd /swift-nio
# swift build
[8/36] Compiling Swift Module 'NIO' (55 sources)

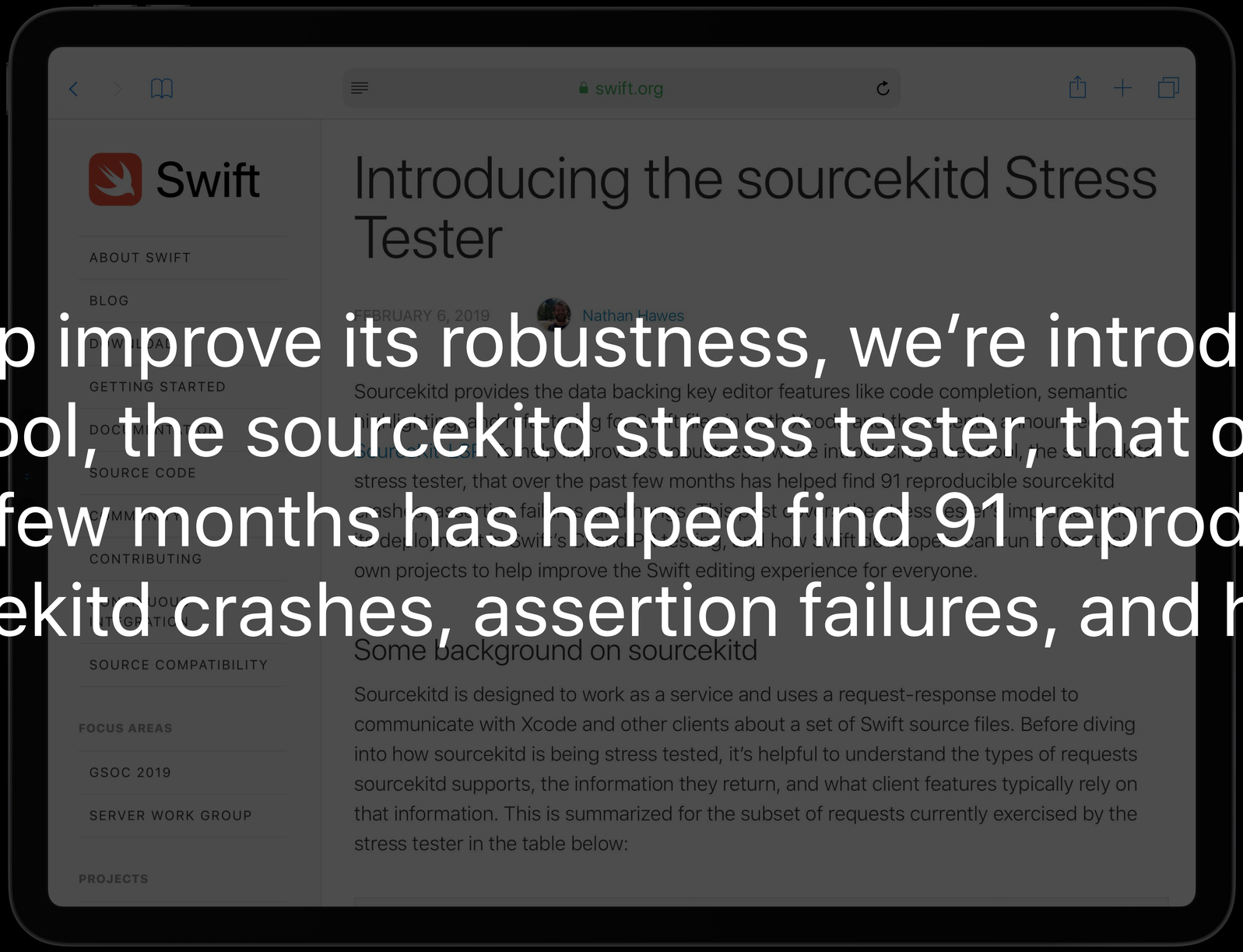

$ docker run -it -rm -v ~/swift-nio:/swift-nio swift
# cd /swift-nio
# swift build
[8/36] Compiling Swift Module 'NIO' (55 sources)


SourceKit

Code completion
Jump-to-definition
Refactoring

SourceKit

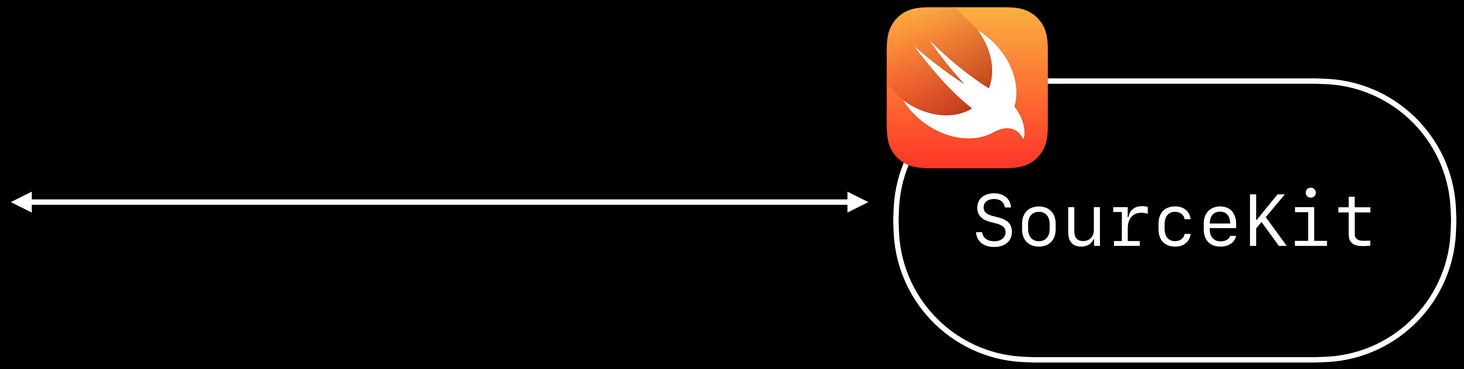

swift.org
Swift
ABOUT SWIFT
BLOG
DOWNLOAD
GETTING STARTED
DOCUMENTATION
SOURCE CODE
COMMUNITY
CONTRIBUTING
CONTINUOUS INTEGRATION
SOURCE COMPATIBILITY
FOCUS AREAS
GSOC 2019
SERVER WORK GROUP
PROJECTS
Introducing the sourcekitd Stress Tester
FEBRUARY 6, 2019
Nathan Hawes
Sourcekitd provides the data backing key editor features like code completion, semantic highlighting, and refactoring for Swift files in both Xcode and the recently announced SourceKit-LSP. To help improve its robustness, we're introducing a new tool, the sourcekitd stress tester, that over the past few months has helped find 91 reproducible sourcekitd crashes, assertion failures, and hangs. This post covers the stress tester's implementation, its deployment in Swift's CI and PR testing, and how Swift developers can run it over their own projects to help improve the Swift editing experience for everyone.
Some background on sourcekitd
Sourcekitd is designed to work as a service and uses a request-response model to communicate with Xcode and other clients about a set of Swift source files. Before diving into how sourcekitd is being stress tested, it's helpful to understand the types of requests sourcekitd supports, the information they return, and what client features typically rely on that information. This is summarized for the subset of requests currently exercised by the stress tester in the table below:



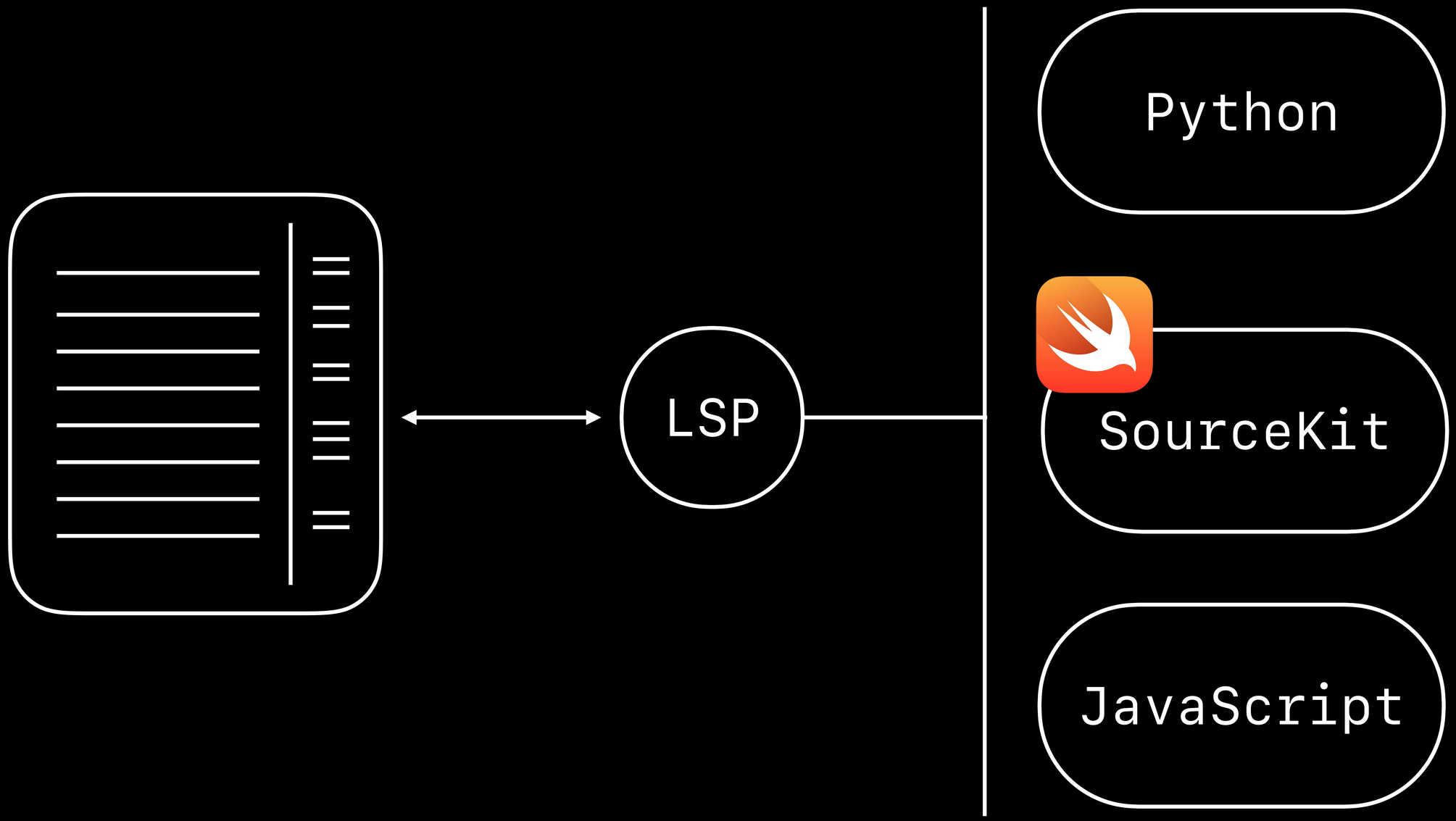
"To help improve its robustness, we're introducing a new tool, the sourcekitd stress tester, that over the past few months has helped find 91 reproducible sourcekitd crashes, assertion failures, and hangs."

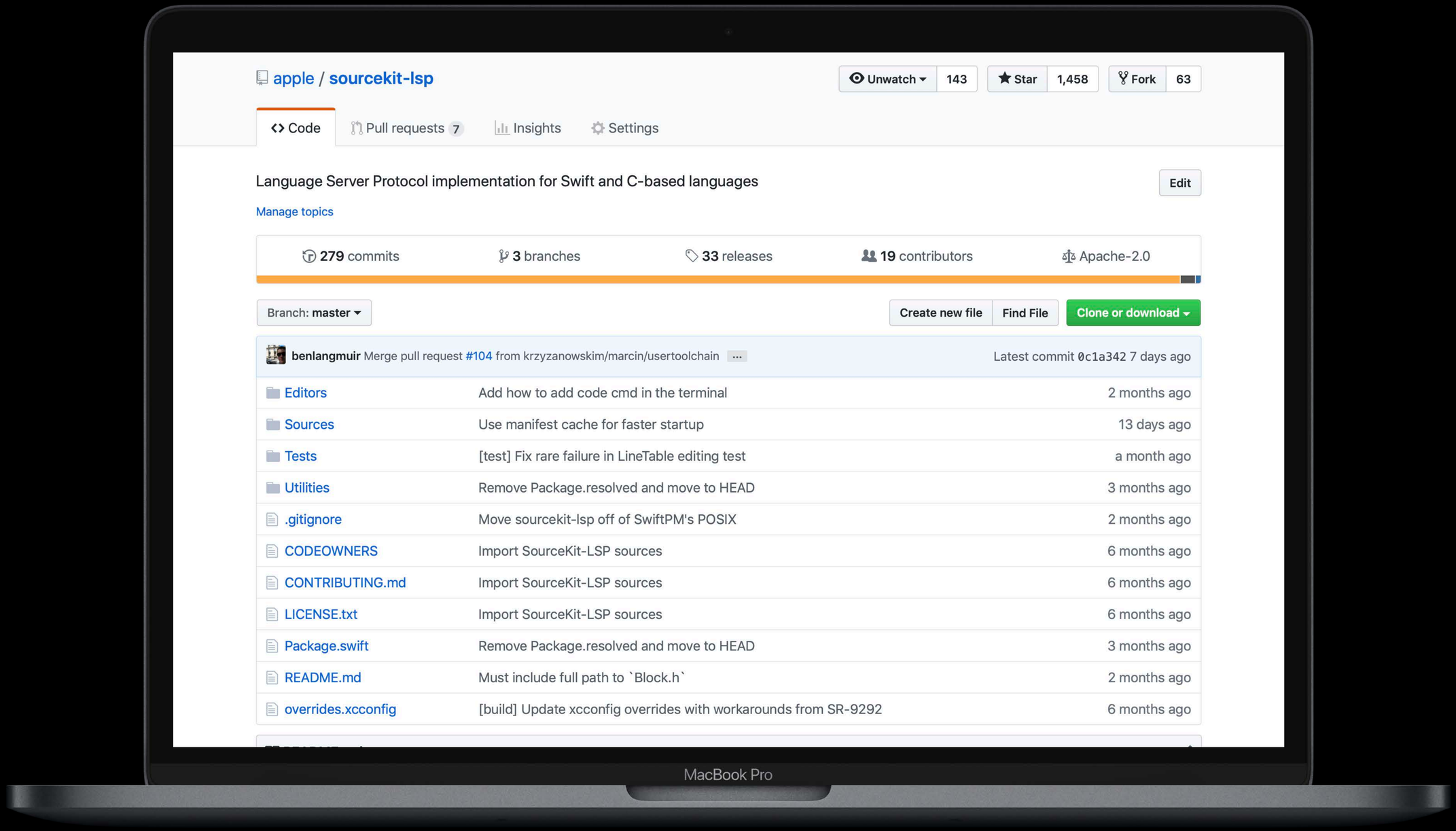
Language Server Protocol

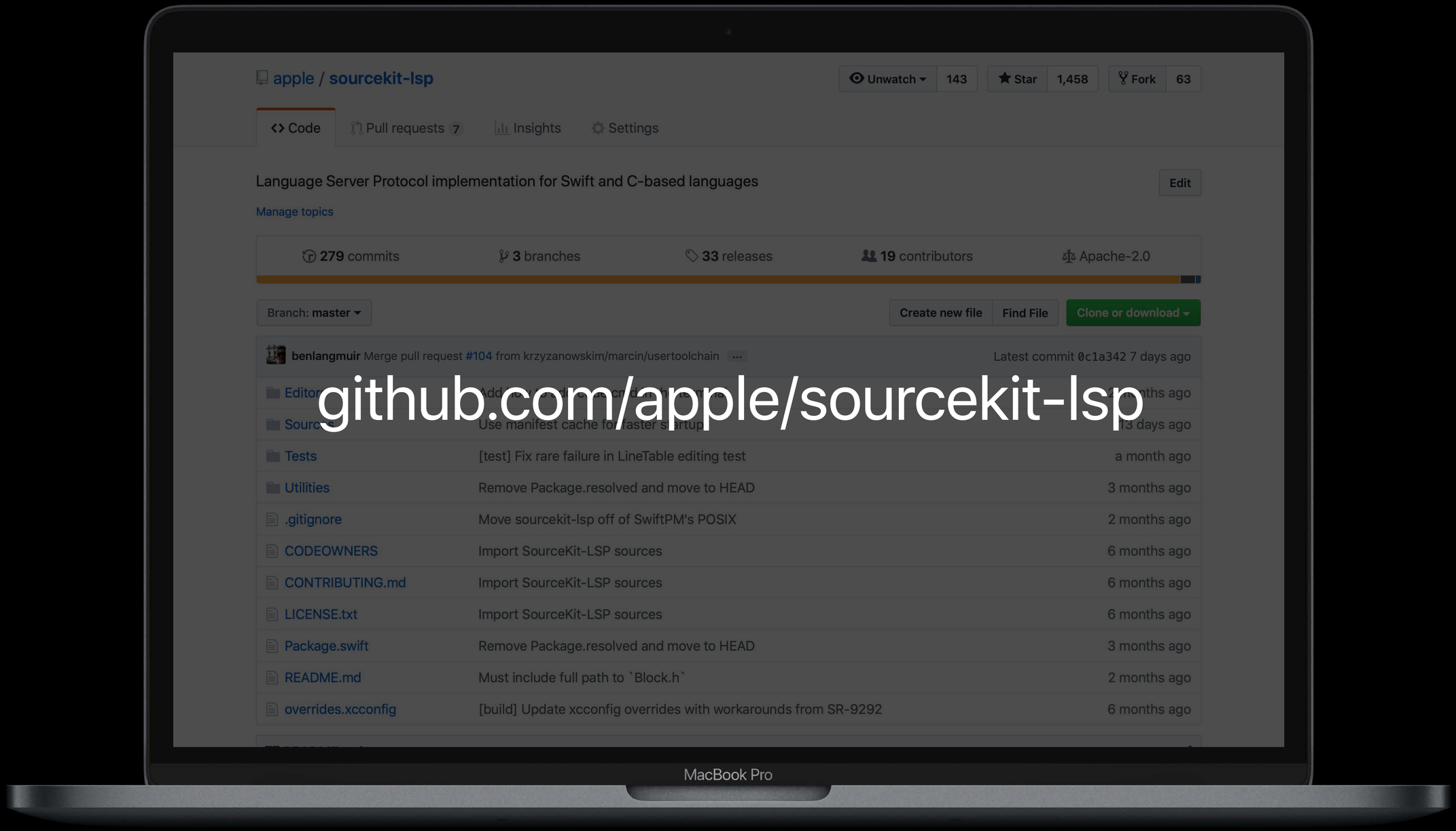

XCODE
SourceKit


SourceKit

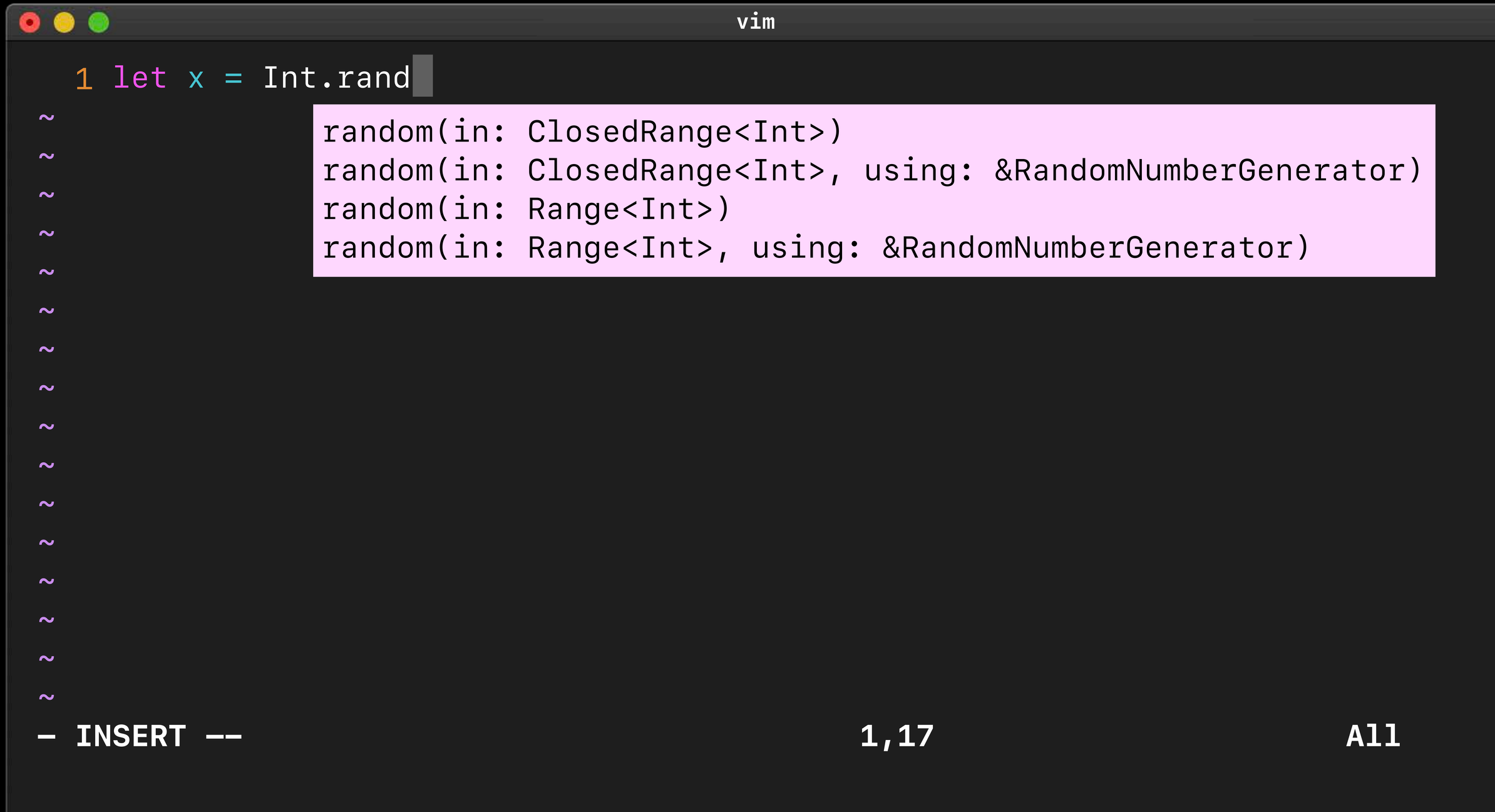

SourceKit


Python
LSP
SourceKit
JavaScript


apple / sourcekit-lsp
Unwatch 143
Star 1,458
Fork 63
Code
Pull requests 7
Insights
Settings
Language Server Protocol implementation for Swift and C-based languages
Edit
Manage topics
279 commits
3 branches
33 releases
19 contributors
Apache-2.0
Branch: master
Create new file
Find File
Clone or download
benlangmuir Merge pull request #104 from krzyzanowskim/marcin/usertoolchain
Latest commit 0c1a342 7 days ago
Editors Add how to add code cmd in the terminal 2 months ago
Sources Use manifest cache for faster startup 13 days ago
Tests [test] Fix rare failure in LineTable editing test a month ago
Utilities Remove Package.resolved and move to HEAD 3 months ago
.gitignore Move sourcekit-lsp off of SwiftPM's POSIX 2 months ago
CODEOWNERS Import SourceKit-LSP sources 6 months ago
CONTRIBUTING.md Import SourceKit-LSP sources 6 months ago
LICENSE.txt Import SourceKit-LSP sources 6 months ago
Package.swift Remove Package.resolved and move to HEAD 3 months ago
README.md Must include full path to `Block.h` 2 months ago
overrides.xcconfig [build] Update xcconfig overrides with workarounds from SR-9292 6 months ago
MacBook Pro


apple / sourcekit-lsp
Unwatch 143
Star 1,458
Fork 63
Code
Pull requests 7
Insights
Settings
Language Server Protocol implementation for Swift and C-based languages
Edit
Manage topics
279 commits
3 branches
33 releases
19 contributors
Apache-2.0
Branch: master
Create new file
Find File
Clone or download
benlangmuir Merge pull request #104 from krzyzanowskim/marcin/usertoolchain
Latest commit 0c1a342 7 days ago
Tests [test] Fix rare failure in LineTable editing test a month ago
Utilities Remove Package.resolved and move to HEAD 3 months ago
.gitignore Move sourcekit-lsp off of SwiftPM's POSIX 2 months ago
CODEOWNERS Import SourceKit-LSP sources 6 months ago
CONTRIBUTING.md Import SourceKit-LSP sources 6 months ago
LICENSE.txt Import SourceKit-LSP sources 6 months ago
Package.swift Remove Package.resolved and move to HEAD 3 months ago
README.md Must include full path to `Block.h` 2 months ago
overrides.xcconfig [build] Update xcconfig overrides with workarounds from SR-9292 6 months ago
github.com/apple/sourcekit-lsp
MacBook Pro


vim
1
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
1,17
All


vim
1 let x = Int.rand
random(in: ClosedRange<Int>)
random(in: ClosedRange<Int>, using: &RandomNumberGenerator)
random(in: Range<Int>)
random(in: Range<Int>, using: &RandomNumberGenerator)
-- INSERT --
1,17
All

Language and Standard Library

Anna Zaks, Swift Team

Implicit returns
(SE-0255)

Generic math functions
(SE-0246)

Default values for
memberwise initializer
(SE-0242)

Static and class subscripts
(SE-0254)

Ordered Collection Diffing
(SE-240)

SIMD refinements
(SE-0229)

Implicit return from
single expressions
(SE-0255)

Key path expressions
as functions
(SE-0249)

Literal initialization
via coercion
(SE-0213)

Identity key path
(SE-0227)

Result type
(SE-0235)

Implicit returns
(SE-0255)

Generic math functions
(SE-0246)

Default values for
memberwise initializer
(SE-0242)

Static and class subscripts
(SE-0254)

Ordered Collection Diffing
(SE-240)

SIMD refinements
(SE-0229)

Property wrappers
(SE-0258)

Key path member lookup
(SE-0252)

Dynamically “callable” types
(SE-0216)

Library evolution for
stable ABIs
(SE-0260)

String interpolation
improvements
(SE-0228)

Opaque result types
(SE-0244)

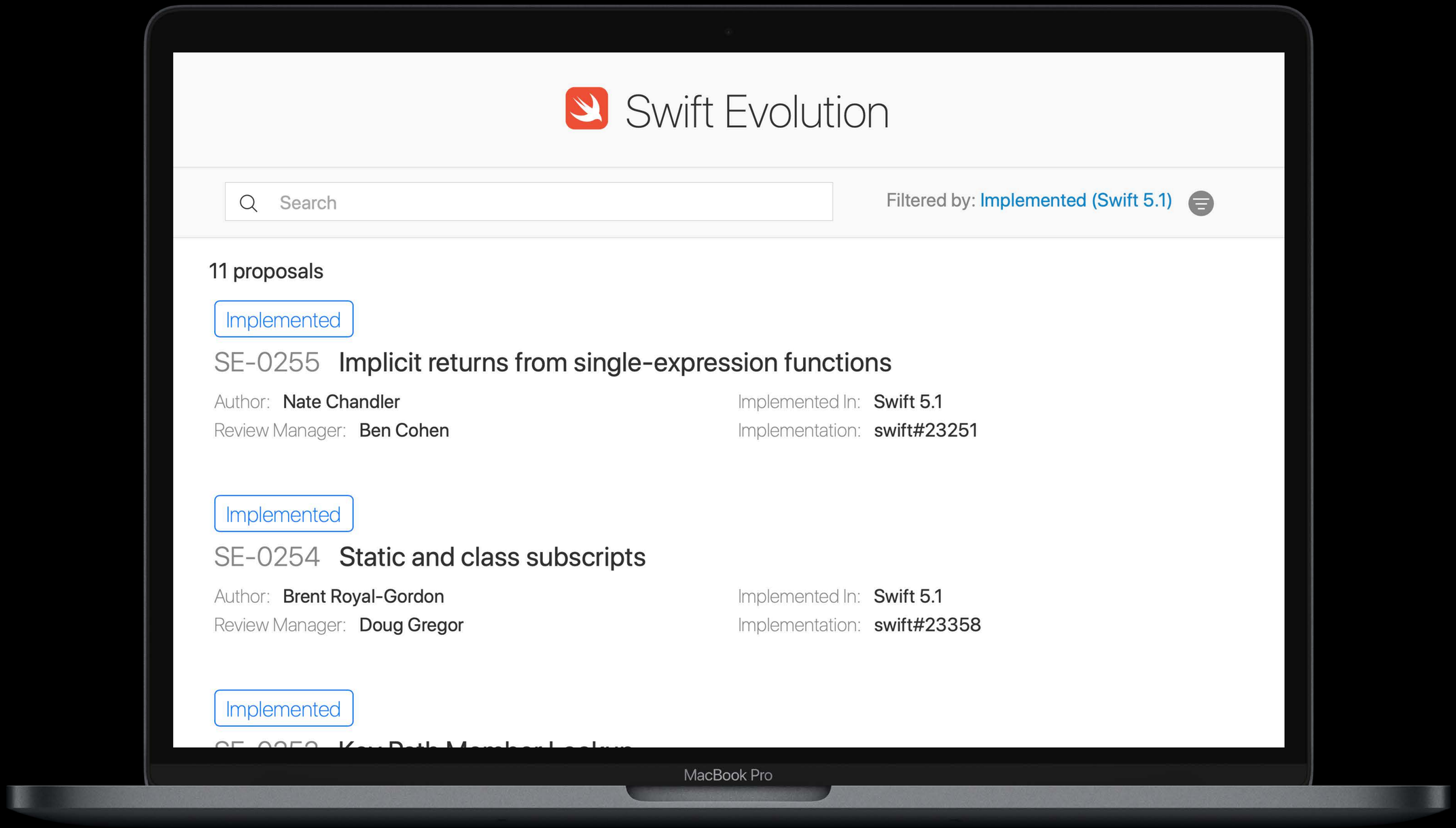
Implicit return from
single expressions
(SE-0255)

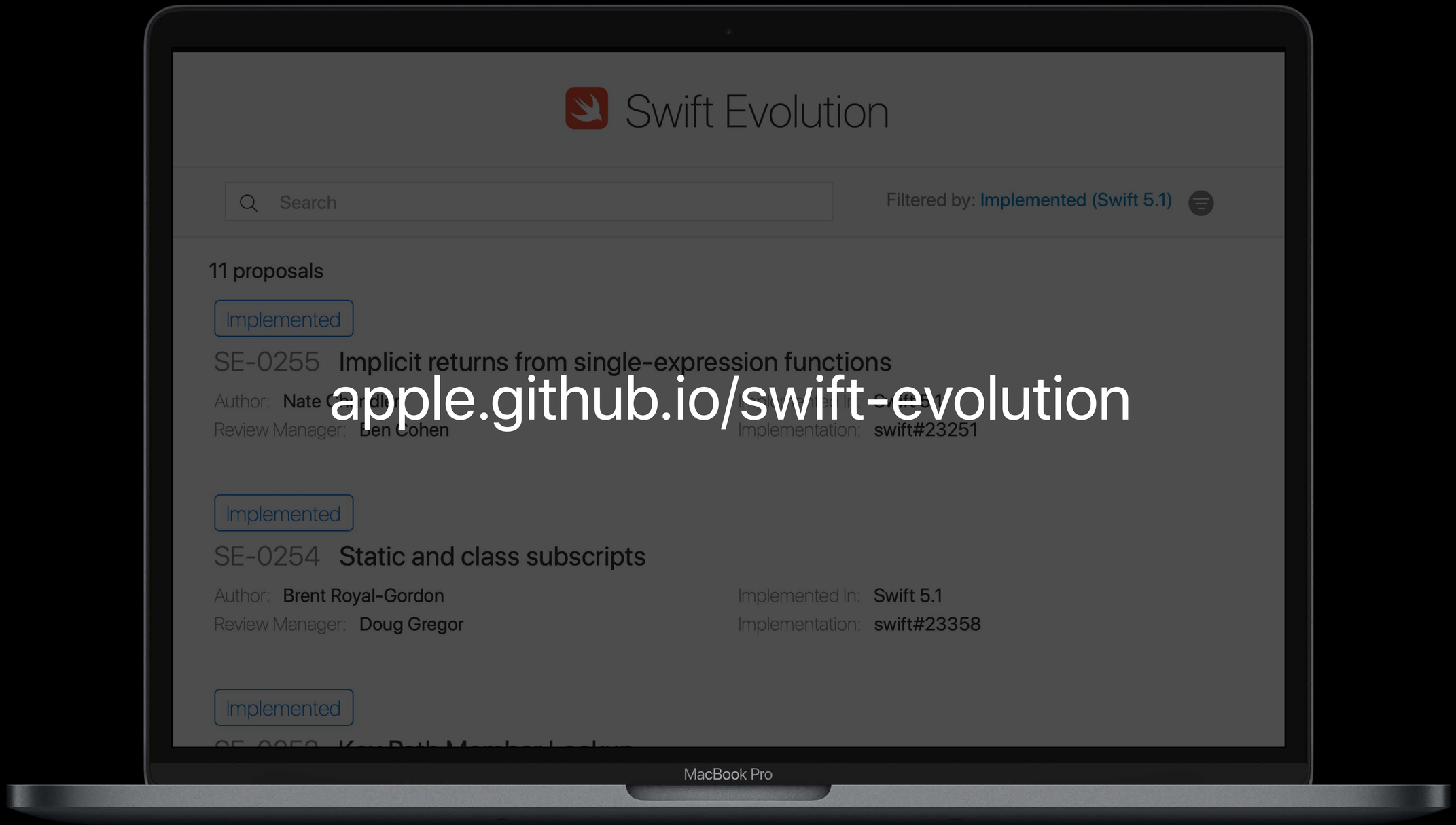
Key path expressions
as functions
(SE-0249)

Literal initialization
via coercion
(SE-0213)

Identity key path
(SE-0227)

Result type
(SE-0235)


Swift Evolution
Search
Filtered by: Implemented (Swift 5.1)
11 proposals
Implemented
SE-0255 Implicit returns from single-expression functions
Author: Nate Chandler
Review Manager: Ben Cohen
Implemented In: Swift 5.1
Implementation: swift#23251
Implemented
SE-0254 Static and class subscripts
Author: Brent Royal-Gordon
Review Manager: Doug Gregor
Implemented In: Swift 5.1
Implementation: swift#23358
Implemented
MacBook Pro


Swift Evolution
Search
Filtered by: Implemented (Swift 5.1)
11 proposals
Implemented
SE-0255 Implicit returns from single-expression functions
Author: Nate Cook
Review Manager: Ben Cohen
Implementation: swift#23251
Implemented
SE-0254 Static and class subscripts
Author: Brent Royal-Gordon
Review Manager: Doug Gregor
Implemented In: Swift 5.1
Implementation: swift#23358
Implemented
apple.github.io/swift-evolution
MacBook Pro

```
// Implicit return from single expressions
// Swift Evolution: SE-0255

struct Rectangle {
    var width = 0.0, height = 0.0
    var area: Double {
        return width * height
    }
}
```

```
// Implicit return from single expressions
// Swift Evolution: SE-0255

struct Rectangle {
    var width = 0.0, height = 0.0
    var area: Double { width * height }
}
```

```swift
// Synthesized default values for the memberwise initializer
// Proposal and implementation by an open source contributor Alejandro Alonso
// Swift Evolution: SE-0242

struct Dog {
    var name = "Generic dog name"
    var age = 0
}
```

```swift
// Synthesized default values for the memberwise initializer
// Proposal and implementation by an open source contributor Alejandro Alonso
// Swift Evolution: SE-0242

struct Dog {
    var name = "Generic dog name"
    var age = 0
}

let boltNewborn = Dog()

let daisyNewborn = Dog(name: "Daisy", age: 0)

let benjiNewborn = Dog(name: "Benji")
```

cannot invoke initializer for type 'Dog' with an argument list of type '(name: String)'

```swift
// Synthesized default values for the memberwise initializer
// Proposal and implementation by an open source contributor Alejandro Alonso
// Swift Evolution: SE-0242

struct Dog {
    var name = "Generic dog name"
    var age = 0
}

let boltNewborn = Dog()

let daisyNewborn = Dog(name: "Daisy", age: 0)

let benjiNewborn = Dog(name: "Benji")
```

SIMD — A Better API for Vector Programming

Swift Evolution: SE-0229

Types for fixed-size SIMD vectors and matrices:

```
SIMD2<T>, SIMD3<T>, SIMD4<T>, SIMD8<T>, SIMD16<T>, SIMD32<T>, SIMD64<T>
```

Most standard integer and floating-point types can be used as element types

```
// SIMD Vectors API

// Initialize from array literals
let x: SIMD4<Int> = [1,2,3,4]
let y: SIMD4<Int> = [3,2,1,0]

// Pointwise equality, inequality, and ordered comparisons
// Return SIMDMask type
let gr = x .> y
// gr : SIMDMask<SIMD4<Int>>(false, false, true, true)

// Boolean operations on SIMDMasks
let lteq = .!gr
// lteq : SIMDMask<SIMD4<Int>>(true, true, false, false)
```

```
// SIMD Vectors API

// Initialize from array literals
let x: SIMD4<Int> = [1,2,3,4]
let y: SIMD4<Int> = [3,2,1,0]

// Pointwise equality, inequality, and ordered comparisons
// Return SIMDMask type
let gr = x .> y
// gr : SIMDMask<SIMD4<Int>>(false, false, true, true)

// Boolean operations on SIMDMasks
let lteq = .!gr
// lteq : SIMDMask<SIMD4<Int>>(true, true, false, false)
```

```
// SIMD Vectors API

// Initialize from array literals
let x: SIMD4<Int> = [1,2,3,4]
let y: SIMD4<Int> = [3,2,1,0]

// Pointwise equality, inequality, and ordered comparisons
// Return SIMDMask type
let gr = x .> y
// gr : SIMDMask<SIMD4<Int>>(false, false, true, true)

// Boolean operations on SIMDMasks
let lteq = .!gr
// lteq : SIMDMask<SIMD4<Int>>(true, true, false, false)
```

```
// SIMD Vectors API

// Initialize from array literals
let x: SIMD4<Int> = [1,2,3,4]
let y: SIMD4<Int> = [3,2,1,0]

// Pointwise equality, inequality, and ordered comparisons
// Return SIMDMask type
let gr = x .> y
// gr : SIMDMask<SIMD4<Int>>(false, false, true, true)

// Boolean operations on SIMDMasks
let lteq = .!gr
// lteq : SIMDMask<SIMD4<Int>>(true, true, false, false)
```

```
// SIMD Vectors API

// Initialize from array literals
let x: SIMD4<Int> = [1,2,3,4]
let y: SIMD4<Int> = [3,2,1,0]

// Pointwise equality, inequality, and ordered comparisons
// Return SIMDMask type
let gr = x .> y
// gr : SIMDMask<SIMD4<Int>>(false, false, true, true)

// Boolean operations on SIMDMasks
let lteq = .!gr
// lteq : SIMDMask<SIMD4<Int>>(true, true, false, false)
```

New Design for String Interpolation

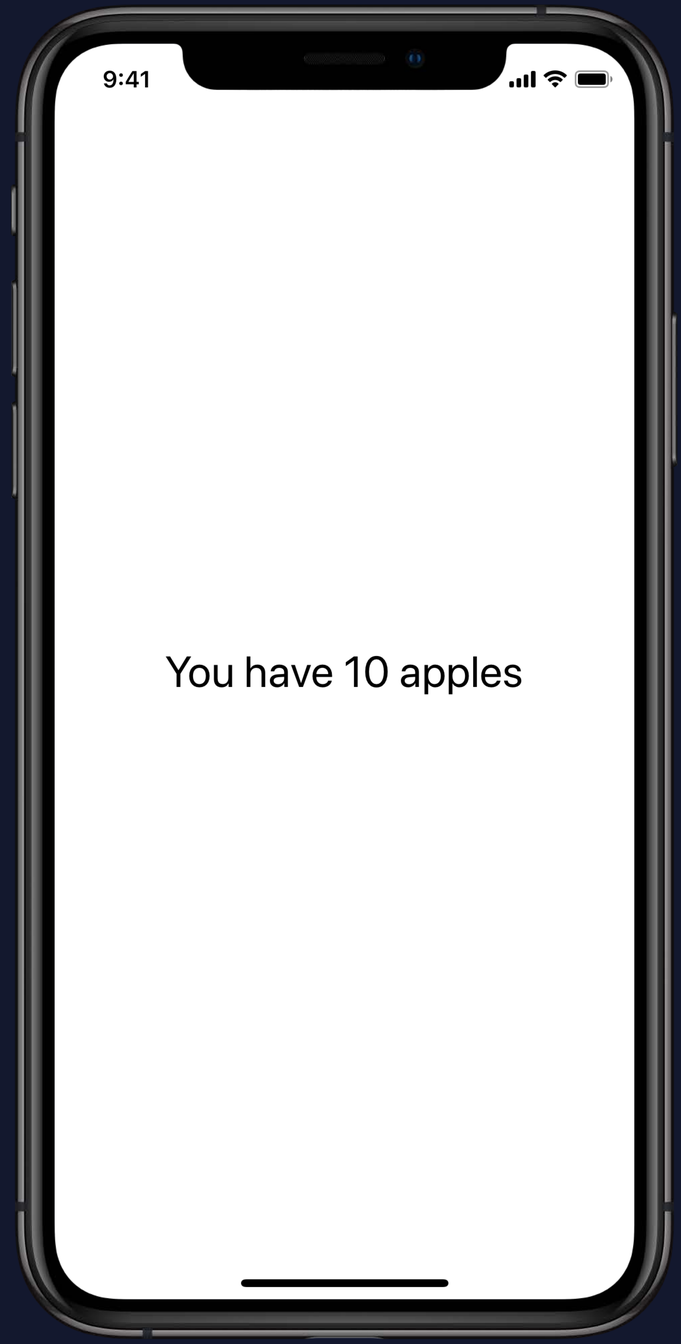
Swift Evolution: SE-0228

Up to 1.7x faster than Swift 4.2's design

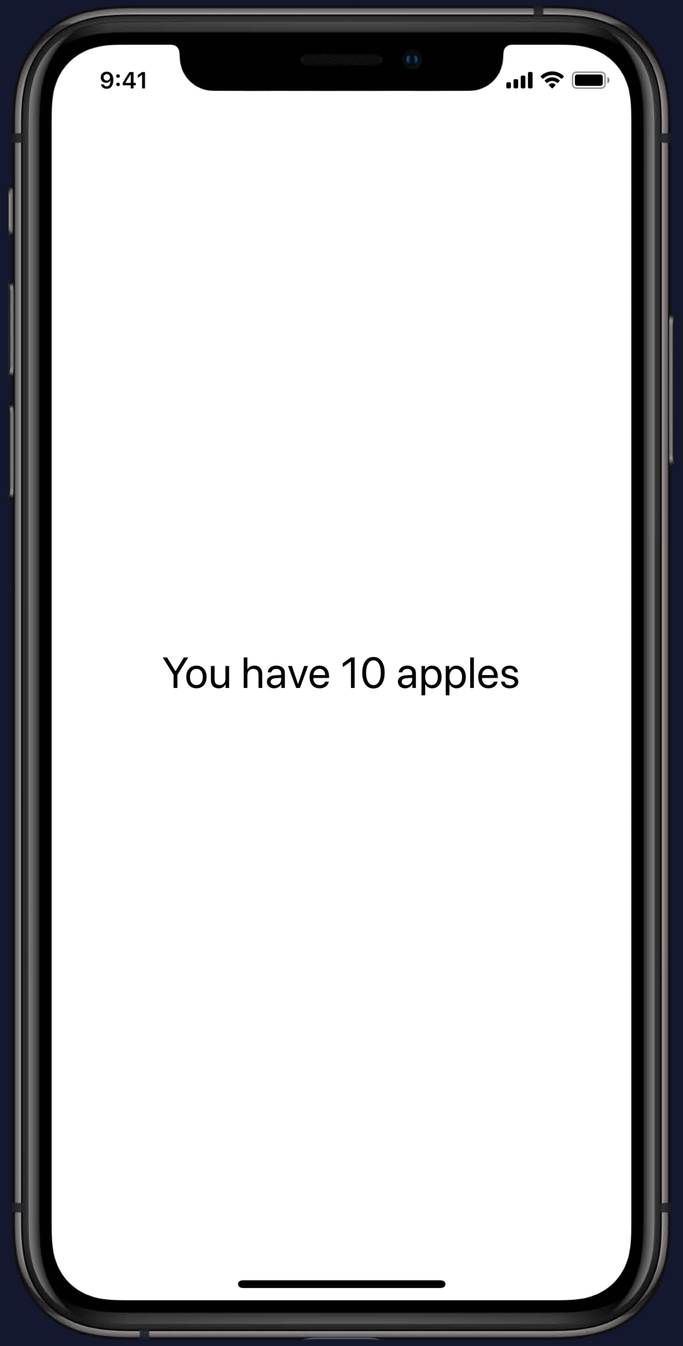
Extend `String` with new interpolation behaviors

Support in your own types by conforming to `ExpressibleByStringInterpolation` protocol

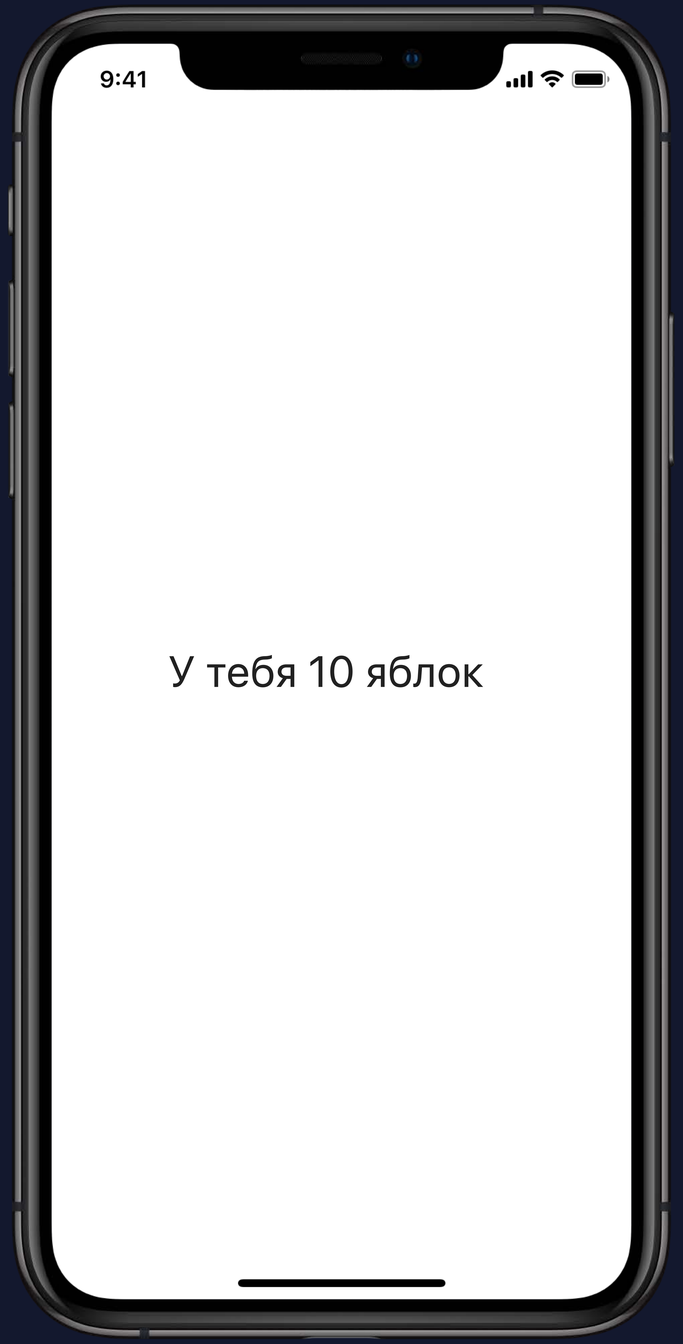
```
let quantity = 10
label.text = "You have \(quantity) apples"
```



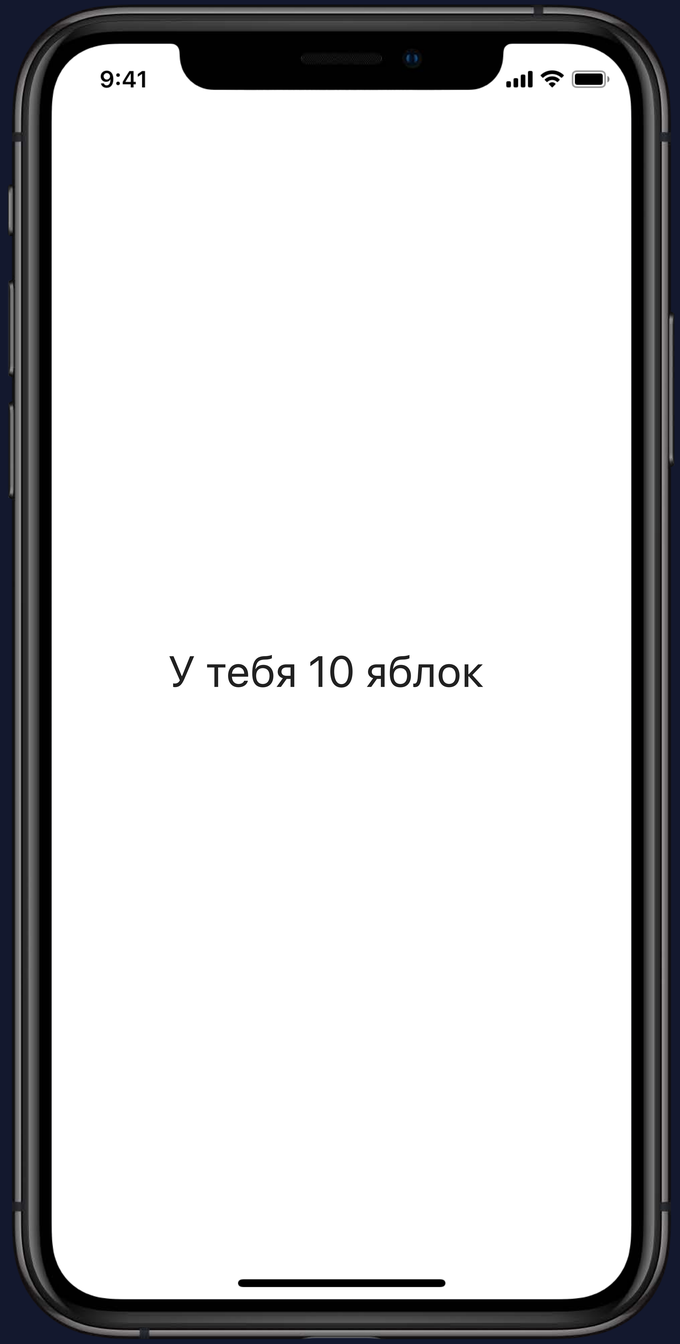
```
let quantity = 10
label.text = NSLocalizedString(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```



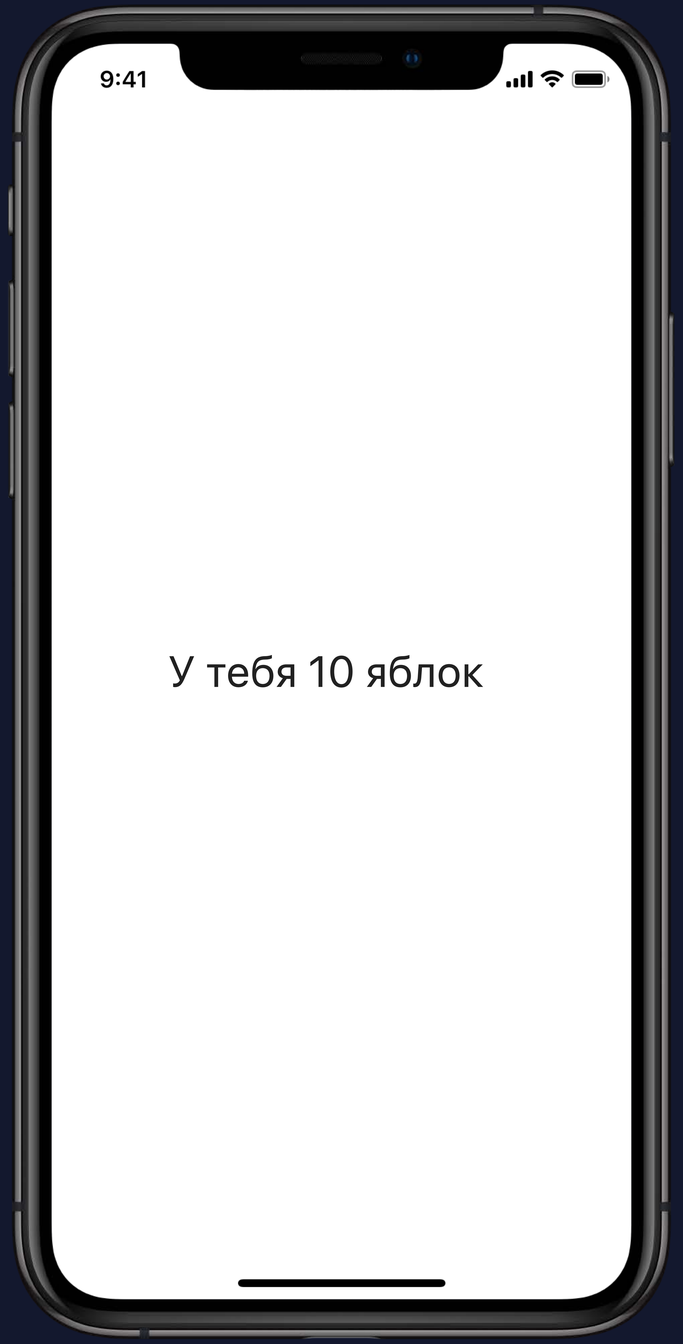
```
let quantity = 10
let formatString = NSLocalizedString(
    "You have %lld apples",
    comment: "Number of apples"
)
label.text = String(format: formatString, quantity)
```



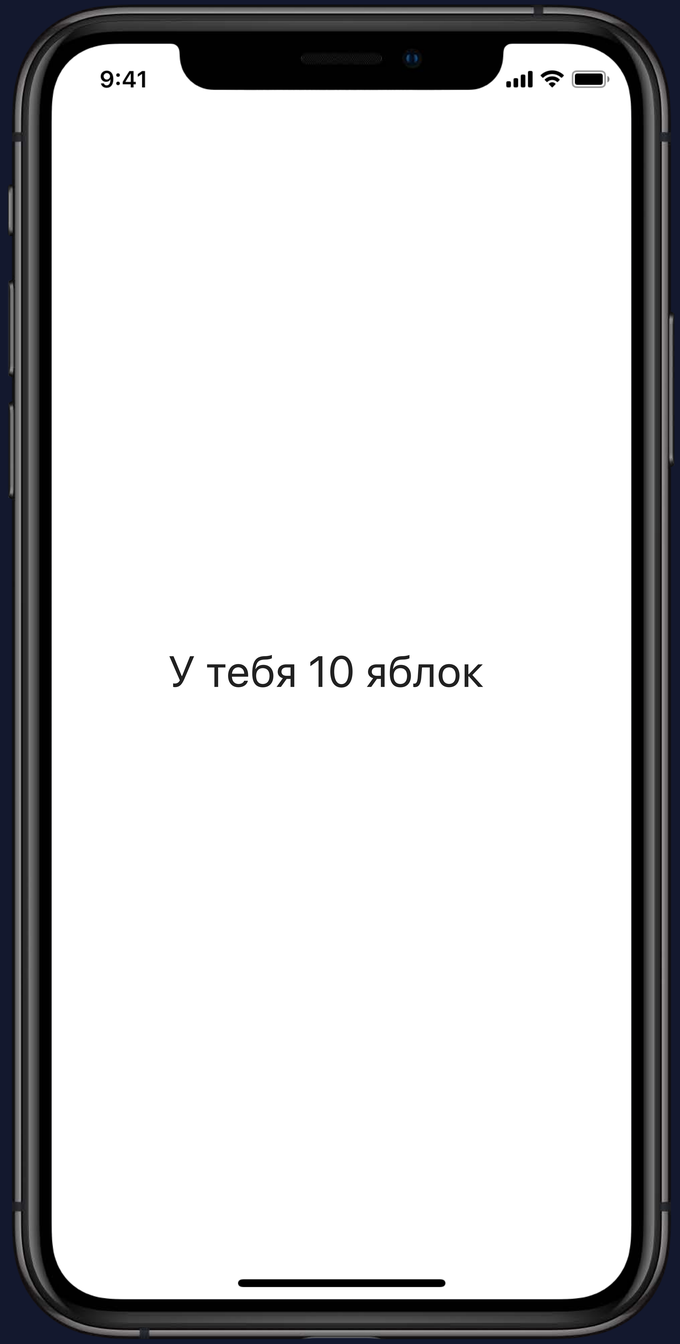
```
let quantity = 10
let formatString = NSLocalizedString(
    "You have %lld apples",
    comment: "Number of apples"
)
label.text = String(format: formatString, quantity)
```



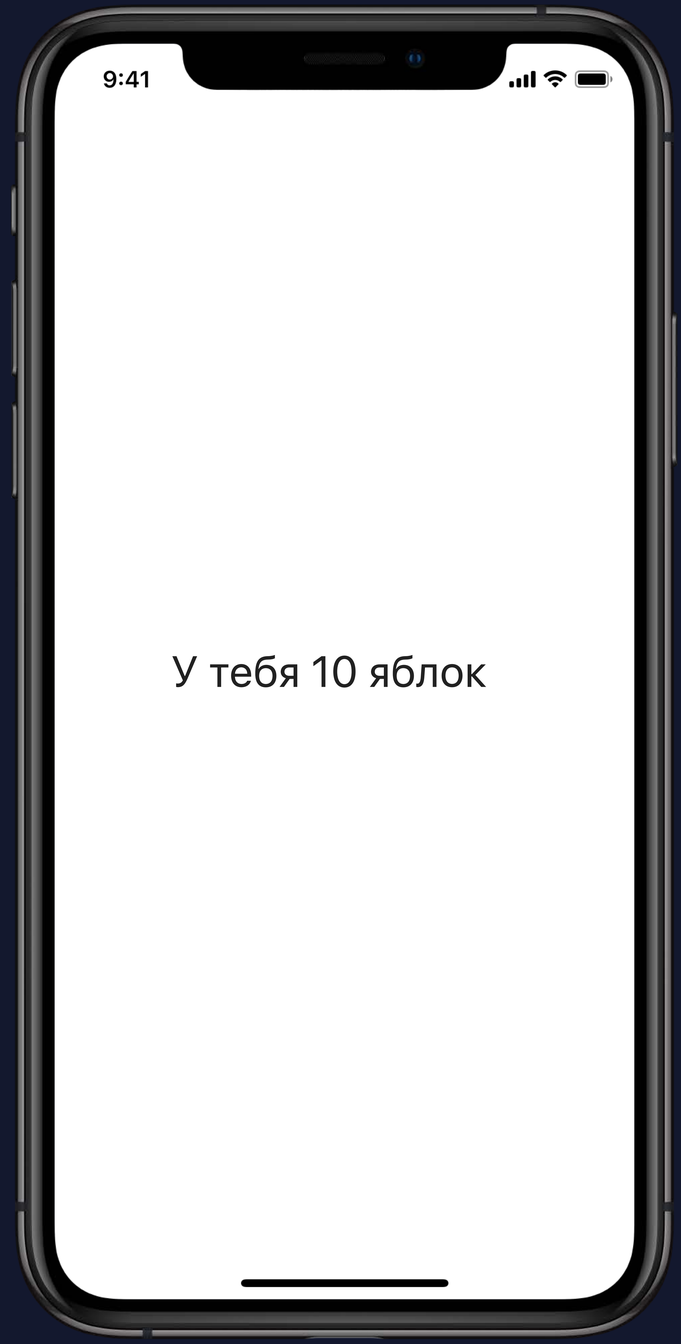
```
let quantity = 10
let formatString = NSLocalizedString(
    "You have %lld apples",
    comment: "Number of apples"
)
label.text = String(format: formatString, quantity)
```



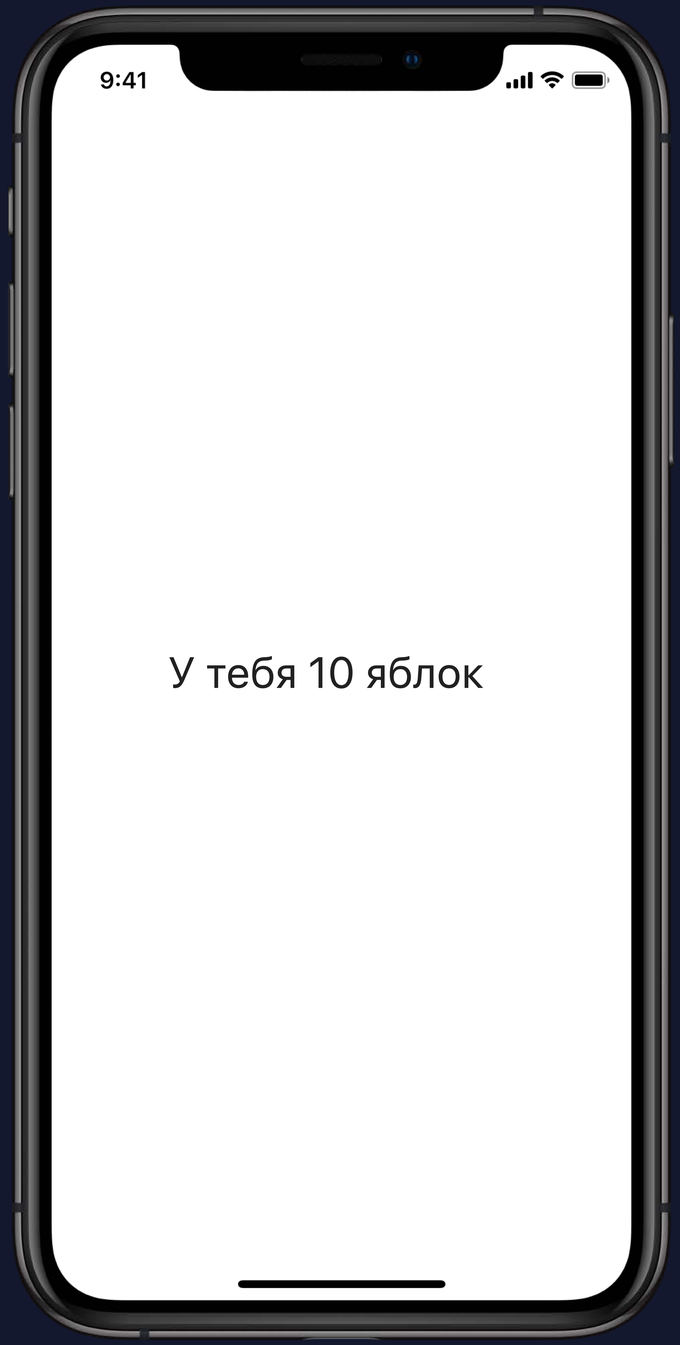
```
let quantity = 10
let formatString = NSLocalizedString(
    "You have %lld apples",
    comment: "Number of apples"
)
label.text = String(format: formatString, quantity)
```



```
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```



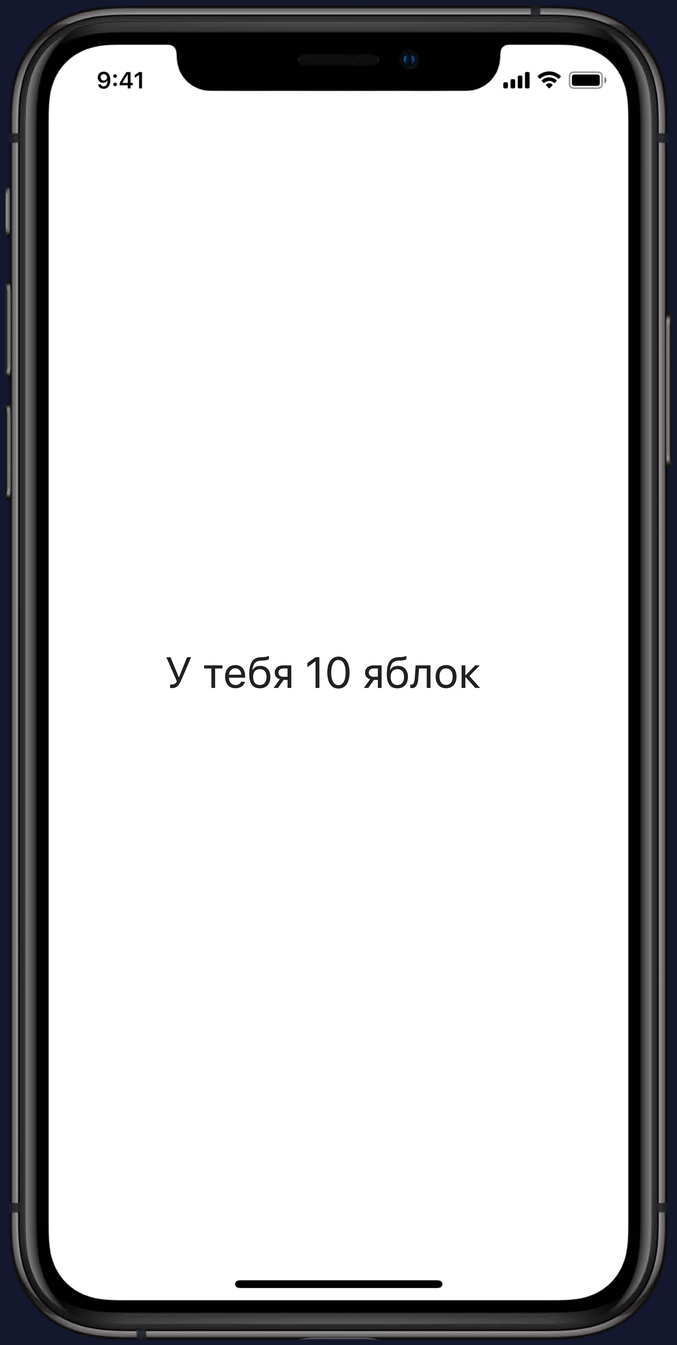
```
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```



```swift
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of appl s"
)
```

as LocalizedStringKey

```swift
// In SwiftUI.framework
public struct Text {
    public init(
        _ key: LocalizedStringKey,
        tableName: String? = nil,
        bundle: Bundle? = nil,
        comment: StaticString? = nil
    )
}
```



```
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```

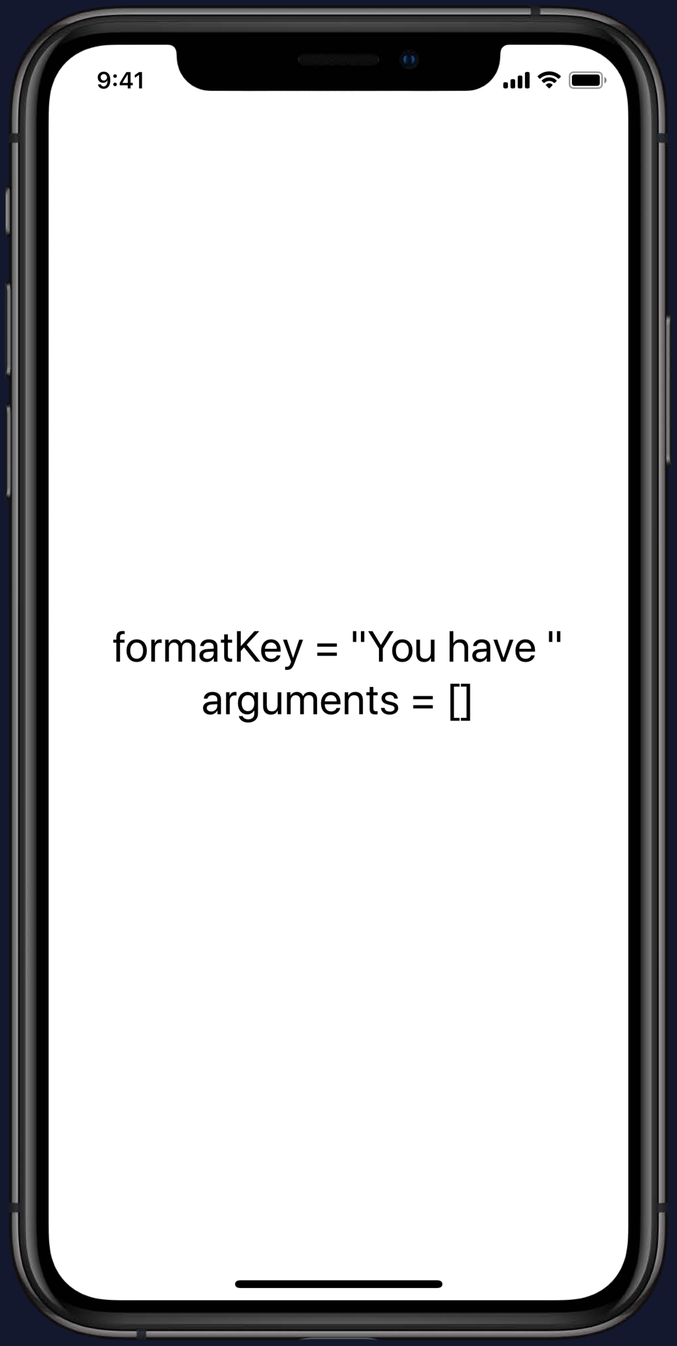
```swift
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```

```swift
// Generated by the Swift compiler
var builder = LocalizedStringKey.StringInterpolation(
    literalCapacity: 16, interpolationCount: 1
)

builder.appendLiteral("You have ")
builder.appendInterpolation(quantity)
builder.appendLiteral(" apples")

LocalizedStringKey(stringInterpolation: builder)
```

```swift
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```


```swift
// Generated by the Swift compiler
var builder = LocalizedStringKey.StringInterpolation(
    literalCapacity: 16, interpolationCount: 1
)

builder.appendLiteral("You have ")
builder.appendInterpolation(quantity)
builder.appendLiteral(" apples")

LocalizedStringKey(stringInterpolation: builder)
```

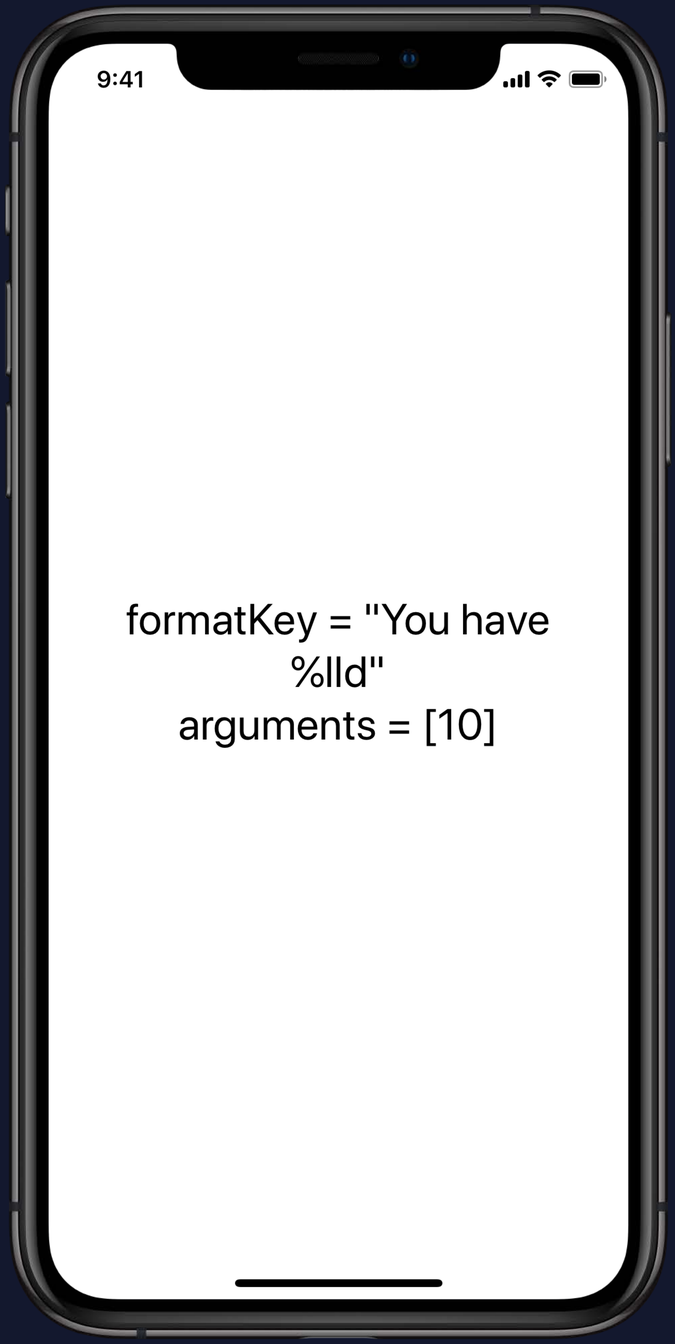
```swift
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```



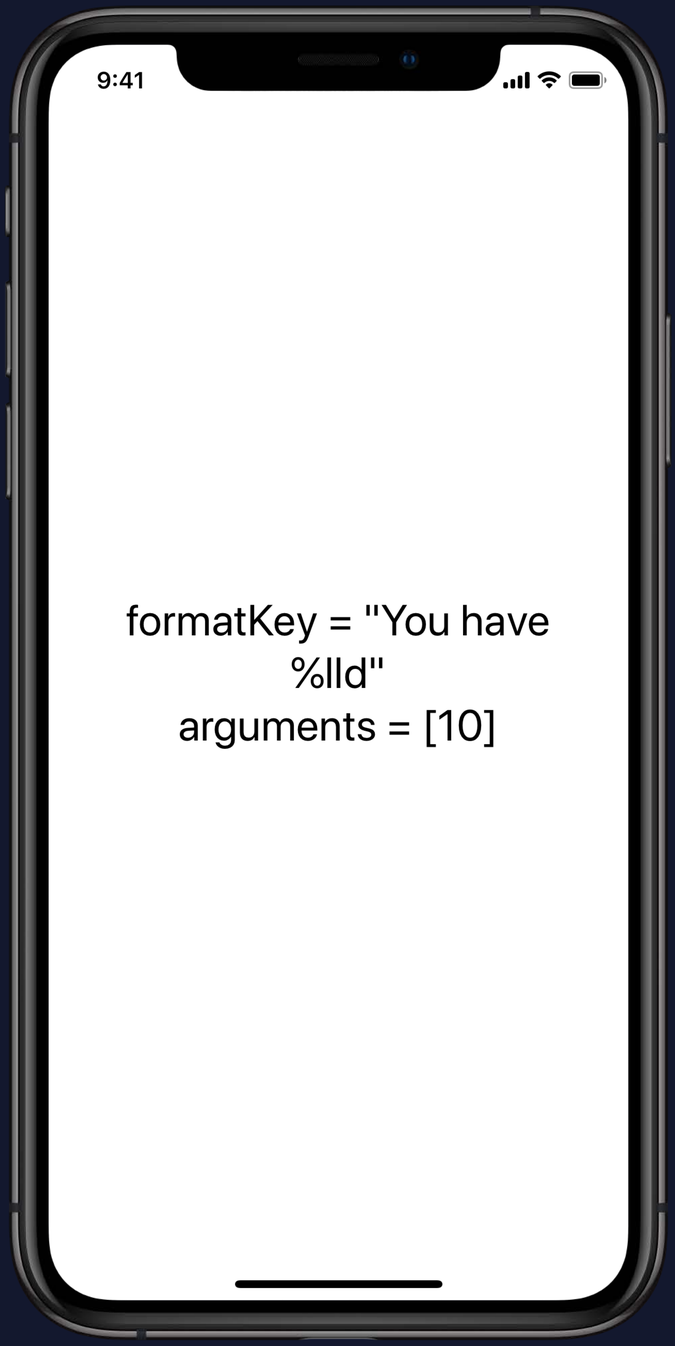

```swift
// Generated by the Swift compiler
var builder = LocalizedStringKey.StringInterpolation(
    literalCapacity: 16, interpolationCount: 1
)

builder.appendLiteral("You have ")
builder.appendInterpolation(quantity)
builder.appendLiteral(" apples")

LocalizedStringKey(stringInterpolation: builder)
```

```swift
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```

```swift
// Generated by the Swift compiler
var builder = LocalizedStringKey.StringInterpolation(
    literalCapacity: 16, interpolationCount: 1
)

builder.appendLiteral("You have ")
builder.appendInterpolation(quantity)
builder.appendLiteral(" apples")

LocalizedStringKey(stringInterpolation: builder)
```



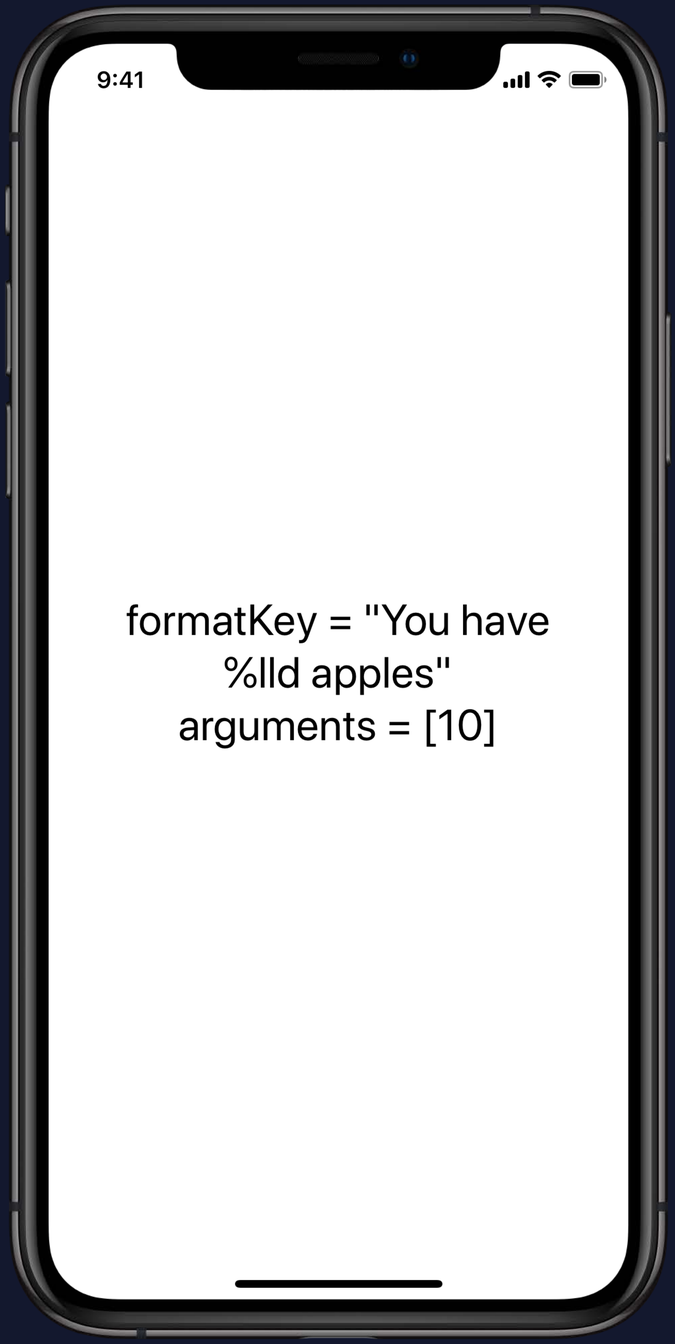
```swift
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```



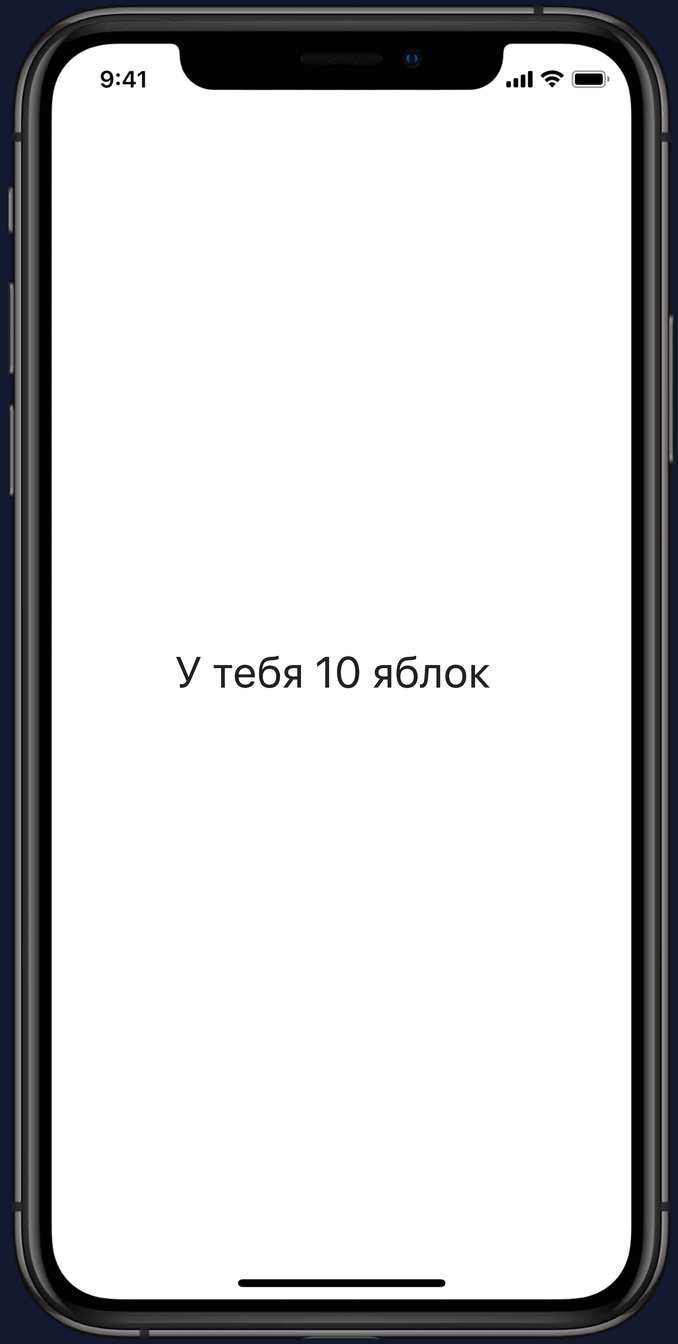

```swift
// Generated by the Swift compiler
var builder = LocalizedStringKey.StringInterpolation(
    literalCapacity: 16, interpolationCount: 1
)

builder.appendLiteral("You have ")
builder.appendInterpolation(quantity)
builder.appendLiteral(" apples")

LocalizedStringKey(stringInterpolation: builder)
```

```swift
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```

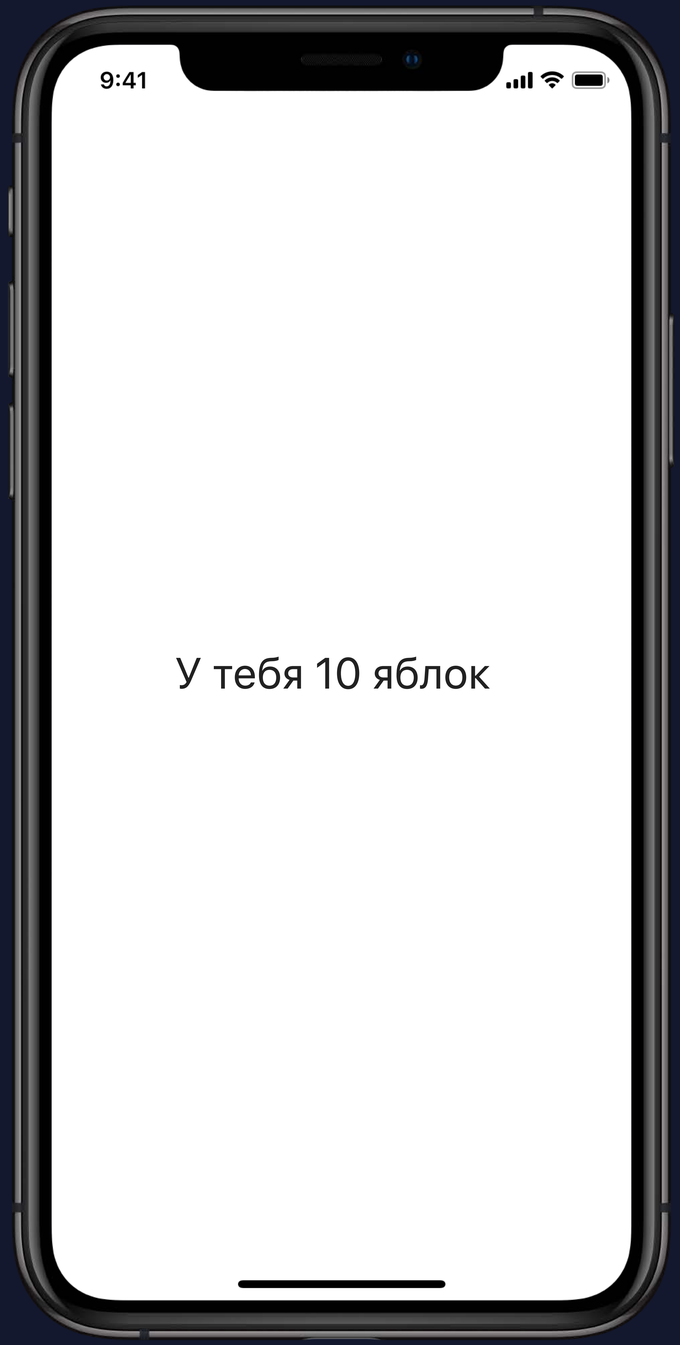
```swift
// Generated by the Swift compiler
var builder = LocalizedStringKey.StringInterpolation(
    literalCapacity: 16, interpolationCount: 1
)

builder.appendLiteral("You have ")
builder.appendInterpolation(quantity)
builder.appendLiteral(" apples")

LocalizedStringKey(stringInterpolation: builder)
```



```
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```



```
let quantity = 10
return Text(
    "You have \(quantity) apples",
    comment: "Number of apples"
)
```



New Design for String Interpolation

Swift Evolution: SE-0228

Framework and package authors are already at work

Read the `ExpressibleByStringInterpolation` documentation to get started

Reasons to Abstract a Return Type

Reasons to Abstract a Return Type

API returns different types conforming to the same protocol

Reasons to Abstract a Return Type

API returns different types conforming to the same protocol

API returns the same type but is leaking implementation details

```
// Shapes Example

protocol Shape { /* ... */ }
struct Square: Shape { /* ... */ }
struct Circle: Shape { /* ... */ }
struct Oval: Shape { /* ... */ }

struct Union<A: Shape, B: Shape>: Shape { /* ... */ }
struct Transformed<S: Shape >: Shape { /* ... */ }
```

```
// API returns different types conforming to the same protocol
// Use Protocol

struct FaceShape {
    …
    var shape: Shape {
        switch faceType {
        case .round:
            return Circle()
        case .square:
            return Square()
        case .diamond:
            return Transformed(Square(), by: .fortyFiveDegrees))
        default:
            return Oval()
        }
    }
}
```

```
// API returns different types conforming to the same protocol
// Use Protocol

struct FaceShape {
    …
    var shape: Shape {
        switch faceType {
        case .round:
            return Circle()
        case .square:
            return Square()
        case .diamond:
            return Transformed(Square(), by: .fortyFiveDegrees))
        default:
            return Oval()
        }
    }
}
```

```
// API returns different types conforming to the same protocol
// Use Protocol

struct FaceShape {
    …
    var shape: Shape {
        switch faceType {
        case .round:
            return Circle()
        case .square:
            return Square()
        case .diamond:
            return Transformed(Square(), by: .fortyFiveDegrees))
        default:
            return Oval()
        }
    }
}
```

```
// API returns the same type but is leaking implementation details


struct EightPointedStar {
    …
    var shape: Union<Square, Transformed<Square>> {
        return Union(Square(), Transformed(Square(), by: .fortyFiveDegrees))
    }
}
```

```
// API returns the same type but is leaking implementation details


struct EightPointedStar {
    …
    var shape: Union<Square, Transformed<Square>> {
        return Union(Square(), Transformed(Square(), by: .fortyFiveDegrees))
    }
}
```

```
// API returns the same type but is leaking implementation details
// Protocol type?

struct EightPointedStar {
    …
    var shape: Shape {
        return Union(Square(), Transformed(Square(), by: .fortyFiveDegrees))
    }
}
```

```
// API returns the same type but is leaking implementation details
// Protocol type?

struct EightPointedStar {
    …
    var shape: Shape {
        return Union(Square(), Transformed(Square(), by: .fortyFiveDegrees))
    }
}
```

Limitations of Returning a Protocol Type

Loses type identity

Does not compose well with the generics system:

- `eightPointedStarOne == eightPointedStarTwo` ✕ Operator '==' cannot be applied
- Returned type cannot have any associated types
- Returned type cannot have requirements that involve `Self`

Disables optimizations

```
// API returns the same type but is leaking implementation details
// Use Opaque Result Types

struct EightPointedStar {
    …
    var shape: some Shape {
        return Union(Square(), Transformed(Square(), by: .fortyFiveDegrees))
    }
}
```

```
// Opaque Result Types
// Compiler enforces that the same type is returned from the implementation

struct EightPointedStar {
    …
    var shape: some Shape {
        if symmetrical {
            return Union(Square(), Transformed(Square(), by: .fortyFiveDegrees))
        } else {
            return Transformed(Square(), by: .twentyDegrees)
        }
    }
}
```

Function returns opaque result type but the return statements do not have matching types

Opaque Result Types

Swift Evolution: SE-0244

Use when API returns the same type but is leaking implementation details

Requires new Swift runtime support

- Available on macOS Catalina, iOS 13, tvOS 13, watchOS 6 and later
- Guard uses with availability checking when deploying to earlier OS releases

Property Wrapper Types

Reuse code for property access patterns

- Lazily initialized values
- Thread-local storage
- Copy on write objects
- Data dependencies in SwiftUI
- User defaults

```swift
static var usesTouchID: Bool {
    get {
        return UserDefaults.standard.bool(forKey: "USES_TOUCH_ID")
    }
    set {
        UserDefaults.standard.set(newValue, forKey: "USES_TOUCH_ID")
    }
}

static var isLoggedIn: Bool {
    get {
        return UserDefaults.standard.bool(forKey: "LOGGED_IN")
    }
    set {
        UserDefaults.standard.set(newValue, forKey: "LOGGED_IN")
    }
}
```

```swift
// The purpose of property wrappers is to wrap a property, specify its access patterns

@propertyWrapper
struct UserDefault<T> {
  let key: String
  let defaultValue: T
  init(_ key: String, defaultValue: T) {
    …
    UserDefaults.standard.register(defaults: [key: defaultValue])
  }
  var value: T {
    get {
      return UserDefaults.standard.object(forKey: key) as? T ?? defaultValue
    }
    set {
      UserDefaults.standard.set(newValue, forKey: key)
    }
  }
}
```

```swift
// The purpose of property wrappers is to wrap a property, specify its access patterns

@propertyWrapper
struct UserDefault<T> {
  let key: String
  let defaultValue: T
  init(_ key: String, defaultValue: T) {
    …
    UserDefaults.standard.register(defaults: [key: defaultValue])
  }
  var value: T {
    get {
      return UserDefaults.standard.object(forKey: key) as? T ?? defaultValue
    }
    set {
      UserDefaults.standard.set(newValue, forKey: key)
    }
  }
}
```

```swift
// The purpose of property wrappers is to wrap a property, specify its access patterns

@propertyWrapper
struct UserDefault<T> {
  let key: String
  let defaultValue: T
  init(_ key: String, defaultValue: T) {
    …
    UserDefaults.standard.register(defaults: [key: defaultValue])
  }
  var value: T {
    get {
      return UserDefaults.standard.object(forKey: key) as? T ?? defaultValue
    }
    set {
      UserDefaults.standard.set(newValue, forKey: key)
    }
  }
}
```

```swift
// The purpose of property wrappers is to wrap a property, specify its access patterns

@propertyWrapper
struct UserDefault<T> {
  let key: String
  let defaultValue: T
  init(_ key: String, defaultValue: T) {
    …
    UserDefaults.standard.register(defaults: [key: defaultValue])
  }
  var value: T {
    get {
      return UserDefaults.standard.object(forKey: key) as? T ?? defaultValue
    }
    set {
      UserDefaults.standard.set(newValue, forKey: key)
    }
  }
}
```

```
// The purpose of property wrappers is to wrap a property, specify its access patterns

@propertyWrapper
struct UserDefault<T> {
  let key: String
  let defaultValue: T
  init(_ key: String, defaultValue: T) {
    …
    UserDefaults.standard.register(defaults: [key: defaultValue])
  }
  var value: T {
    get {
      return UserDefaults.standard.object(forKey: key) as? T ?? defaultValue
    }
    set {
      UserDefaults.standard.set(newValue, forKey: key)
    }
  }
}
```

Allows writing: `@UserDefault(…) var x: Bool`

```
// Using UserDefault property wrapper to declare and access properties

@UserDefault("USES_TOUCH_ID", defaultValue: false)
static var usesTouchID: Bool

@UserDefault("LOGGED_IN", defaultValue: false)
static var isLoggedIn: Bool
```

```
// Using UserDefault property wrapper to declare and access properties

@UserDefault("USES_TOUCH_ID", defaultValue: false)
static var usesTouchID: Bool

@UserDefault("LOGGED_IN", defaultValue: false)
static var isLoggedIn: Bool
```

```
// Using UserDefault property wrapper to declare and access properties

@UserDefault("USES_TOUCH_ID", defaultValue: false)
static var usesTouchID: Bool

@UserDefault("LOGGED_IN", defaultValue: false)
static var isLoggedIn: Bool

if !isLoggedIn && usesTouchID {
    !authenticateWithTouchID()
}
```

Property Wrapper Types

Swift Evolution: SE-0258

Wrapper types define custom access patterns

Property can adopt by adding an attribute to its declaration

DSLs (Domain Specific Languages)

```
<html>
<head>
  <title>
    Welcome Message
  </title>
</head>
<body>
    Welcome to WWDC19!
</body>
</html>
```

```
INSERT INTO attendee
VALUES
    ('Welcome to WWDC19!', name);
```

```
\begin{document}
Welcome to WWDC19!
\end{document}
```

```
digraph G { Welcome to->WWDC19! }
```

```
{
  "message": "Welcome to WWDC19!"
}
```

Tools Support is Vital

```
<html>
    <head>
        <title>Jessica's WWDC19 Blog</title>
    </head>
    <body>
        <h2>Welcome to Jessica's WWDC19 Blog!</h2>
        <br>
        Check out the talk schedule and latest news from
        <a href="https://developer.apple.com/wwdc19/">the source</a>
    </body>
</html>
```

Tools Support is Vital

```
"""<html>
        <head>
            <title>\(name)'s WWDC19 Blog</title>
        </head>
        <body>
            <h2>Welcome to \(name)'s WWDC19 Blog!
            <br>
            Check out the talk schedule and latest news from
            <a href="https://developer.apple.com/wwdc19/">the source</a>
        </body>
    </html>
"""
```

Tools Support is Vital

```
"""<html>
        <head>
            <title>\(name)'s WWDC19 Blog</title>
        </head>
        <body>
            <h2>Welcome to \(name)'s WWDC19 Blog!
            <br>
            Check out the talk schedule and latest news from
            <a href="https://developer.apple.com/wwdc19/">the source</a>
        </body>
    </html>
"""
```

Tools Support is Vital

```
"""<html>
        <head>
            <title>\(name)'s WWDC19 Blog</title>
        </head>
        <body>
            <h2>Welcome to \(name)'s WWDC19 Blog!
            <br>
            Check out the talk schedule and latest news from
            <a href="https://developer.apple.com/wwdc19/">the source</a>
        </body>
    </html>
"""
```

Tools Support is Vital

```
"""<html>
        <head>
            <title>\(name)'s WWDC19 Blog</title>
        </head>
        <body>
            <h2>Welcome to \(name)'s WWDC19 Blog!</h2>
            <br>
            Check out the talk schedule and latest news from
            <a href="https://developer.apple.com/wwdc19/">the source</a>
        </body>
    </html>
"""
```

Tools Support is Vital

```
"""<html>
        <head>
            <title>\(name)'s WWDC19 Blog</title>
        </head>
        <body>
            <h2>Welcome to \(name)'s WWDC19 Blog!</h2>
            <br>
            Check out the talk schedule and latest news from
            <a href="https://developer.apple.com/wwdc19/">the source</a>
        </body>
    </html>
"""
```

```
<html>
    <head>
        <title>\(name)'s WWDC19 Blog</title>
    </head>
    <body>
        <h2>Welcome to \(name)'s WWDC19 Blog!</h2>
        <br>
        Check out the talk schedule and latest news from
        <a href="https://developer.apple.com/wwdc19/">the source</a>
    </body>
</html>
```

```
html {
    head {
        title("\(name)'s WWDC19 Blog")
    }
    body {
        h2 { "Welcome to \(name)'s WWDC19 Blog!" }
        br()
        "Check out the talk schedule and latest news from "
        a {
            "the source"
        }.href("https://developer.apple.com/wwdc19/")
    }
}
```

Define Embedded DSLs in Swift

```
html {
    head {
        title("\(name)'s WWDC19 Blog")
    }
    body {
        h2 { "Welcome to \(name)'s WWDC19 Blog!" }
        br()
        "Check out the talk schedule and latest news from "
        a {
            "the source"
        }.href("https://developer.apple.com/wwdc19/")
    }
}
```

```
html {
    head {
        title("\(name)'s WWDC19 Blog")
    }
    body {
        h2 { "Welcome to \(name)'s WWDC19 Blog!" }
        br()
        "Check out the talk schedule and latest news from "
        a {
            "the source"
        }.href("https://developer.apple.com/wwdc19/")
    }
}
```

```
html {
    head {
        title("\(name)'s WWDC19 Blog")
    }
    body {
        h2 { "Welcome to \(name)'s WWDC19 Blog!" }
        br()
        "Check out the talk schedule and latest news from "
        a {
            "the source"
        }.href("https://developer.apple.com/wwdc19/")
        if !loggedIn {
           button {
               "Log in for comments"
           }
           .onclick("triggerLogin()")
       }
    }
}
```

```
html {
    head {
        title("\(name)'s WWDC19 Blog")
    }
    body {
        h2 { "Welcome to \(name)'s WWDC19 Blog!" }
        br()
        "Check out the talk schedule and latest news from "
        a {
            "the source"
        }.href("https://developer.apple.com/wwdc19/")
        if !loggedIn {
            button {
                "Log in for comments"
            }
            .onclick("triggerLogin()")
        }
    }
}
```

```
// Functions that construct the HTML objects

public func html(@HTMLBuilder content: () -> HTML) -> HTML { … }
public func head(@HTMLBuilder content: () -> HTML) -> HTML { … }
public func body(@HTMLBuilder content: () -> HTML) -> HTML { … }
```

```
// Functions that construct the HTML objects

public func html(@HTMLBuilder content: () -> HTML) -> HTML { … }
public func head(@HTMLBuilder content: () -> HTML) -> HTML { … }
public func body(@HTMLBuilder content: () -> HTML) -> HTML { … }
```

```
// Functions that construct the HTML objects

public func html(@HTMLBuilder content: () -> HTML) -> HTML { … }
public func head(@HTMLBuilder content: () -> HTML) -> HTML { … }
public func body(@HTMLBuilder content: () -> HTML) -> HTML { … }
```

Custom attribute backed by `HTMLBuilder` type

Compiler Transforming an `@HTMLBuilder` Closure

```
head {
    meta().charset("UTF-8")
    if cond {
        title("Title 1")
    } else {
        title("Title 2")
    }
}
```

Compiler Transforming an `@HTMLBuilder` Closure

```
head {
    meta().charset("UTF-8")
    if cond {
        title("Title 1")
    } else {
        title("Title 2")
    }
}
```

Compiler Transforming an `@HTMLBuilder` Closure

```
head {
    meta().charset("UTF-8")
    if cond {
        title("Title 1")
    } else {
        title("Title 2")
    }
}
```

```
head {
    let a: HTML = meta().charset("UTF-8")
    let d: HTML
    if cond {
        let b: HTML = title("Title 1")
        d = HTMLBuilder.buildEither(first: b)
    } else {
        let c: HTML = title("Title 2")
        d = HTMLBuilder.buildEither(second: c)
    }
    return HTMLBuilder.buildBlock(a, d)
}
```

Compiler Transforming an `@HTMLBuilder` Closure

```
head {
    meta().charset("UTF-8")
    if cond {
        title("Title 1")
    } else {
        title("Title 2")
    }
}
```

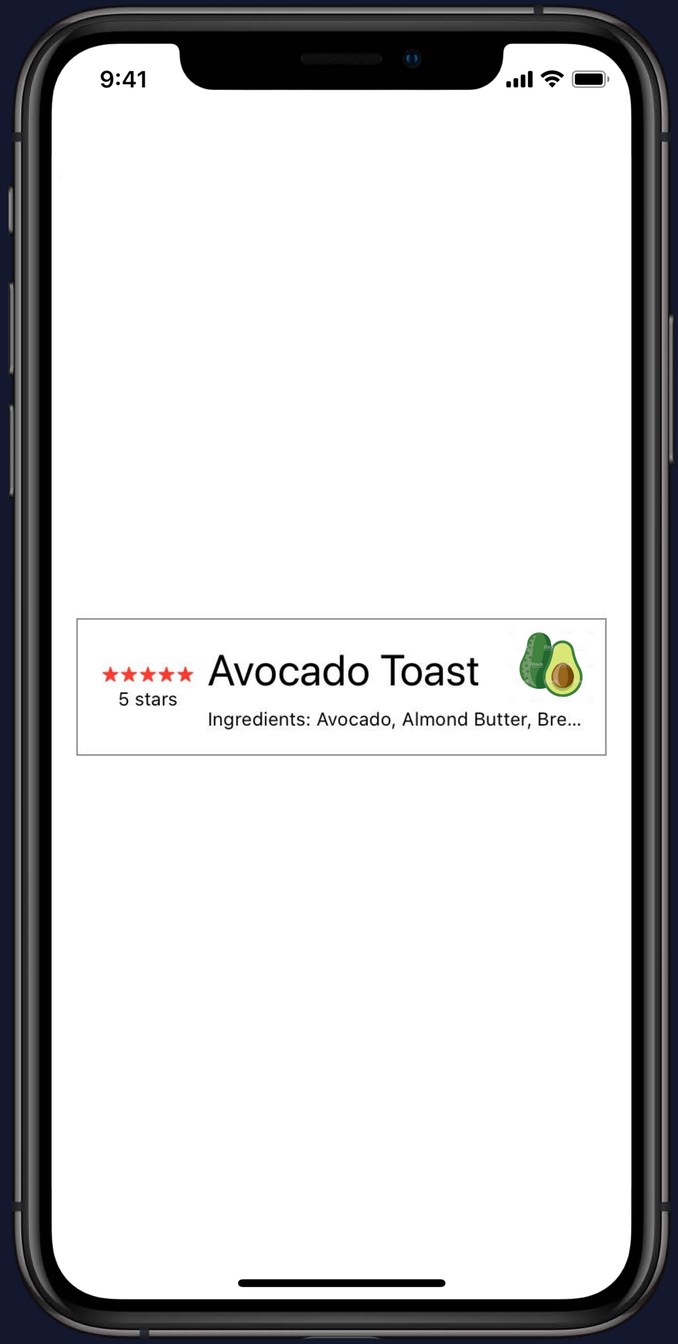
```
head {
    let a: HTML = meta().charset("UTF-8")
    let d: HTML
    if cond {
        let b: HTML = title("Title 1")
        d = HTMLBuilder.buildEither(first: b)
    } else {
        let c: HTML = title("Title 2")
        d = HTMLBuilder.buildEither(second: c)
    }
    return HTMLBuilder.buildBlock(a, d)
}
```

Compiler Transforming an `@HTMLBuilder` Closure

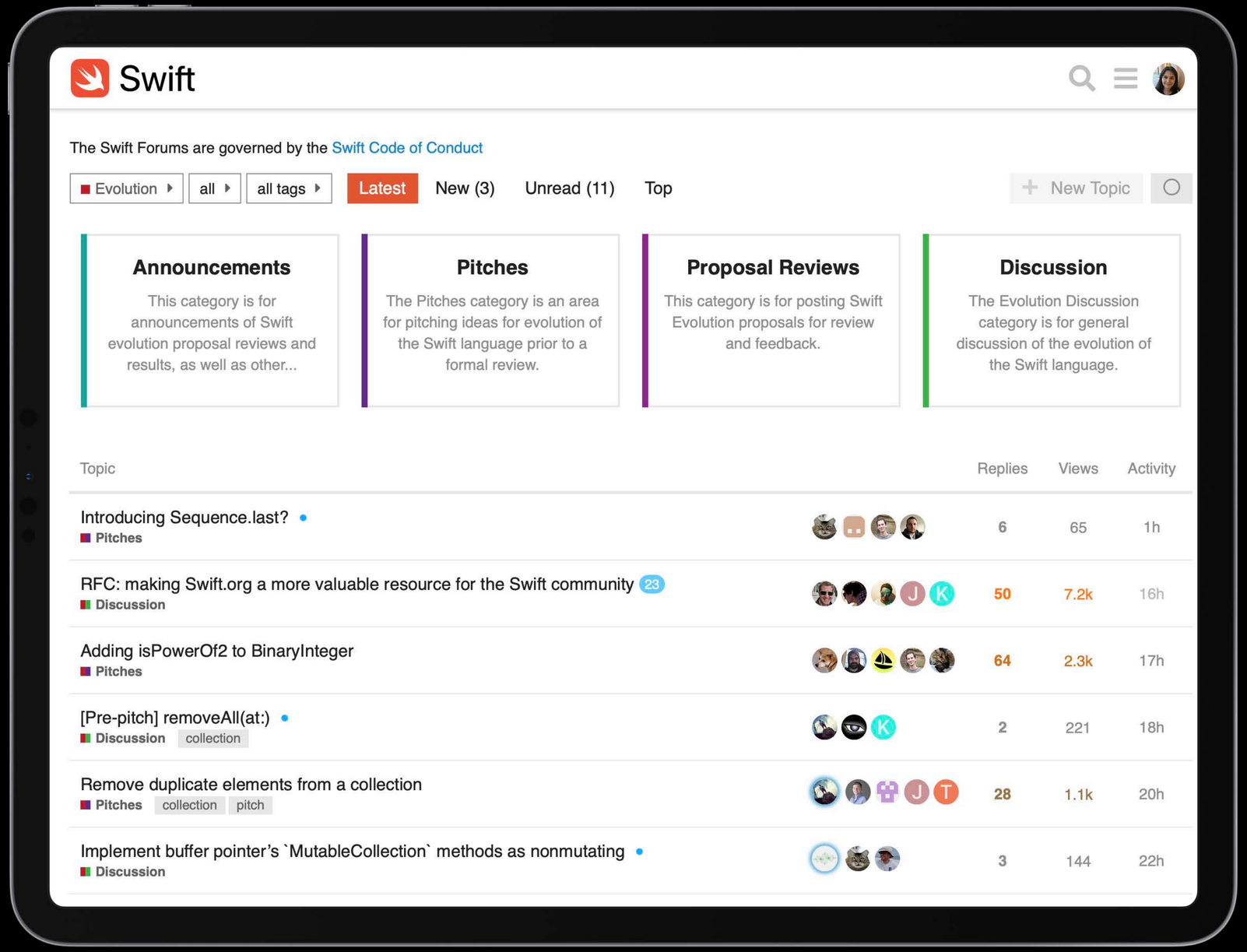
```
head {
    meta().charset("UTF-8")
    if cond {
        title("Title 1")
    } else {
        title("Title 2")
    }
}
```

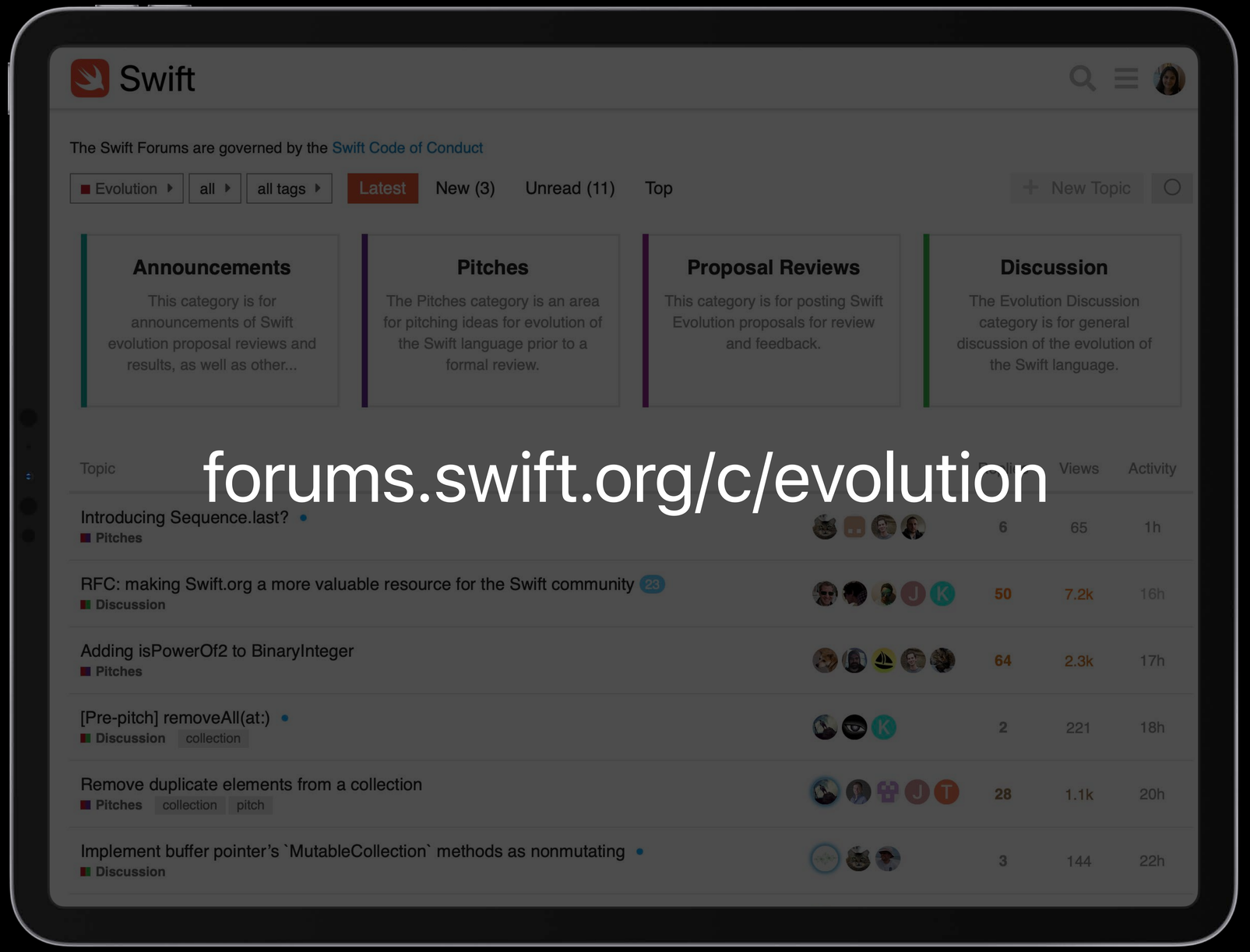
```
head {
    let a: HTML = meta().charset("UTF-8")
    let d: HTML
    if cond {
        let b: HTML = title("Title 1")
        d = HTMLBuilder.buildEither(first: b)
    } else {
        let c: HTML = title("Title 2")
        d = HTMLBuilder.buildEither(second: c)
    }
    return HTMLBuilder.buildBlock(a, d)
}
```

```
// Embedded Swift DSL used in SwiftUI

HStack {
    VStack {
        Text("★★★★★").color(.red)
        Text("5 stars")
    }.font(.caption)

    VStack(alignment: .leading) {
        HStack {
            Text("Avocado Toast").font(.title)
            Spacer()
            Image("avocado_small")
        }
        Text("Ingredients: Avocado, Almond Butter, Bread")
            .font(.caption)
    }
}
```



Swift DSL is waiting for your feedback!


Swift
The Swift Forums are governed by the Swift Code of Conduct
Evolution
all
all tags
Latest
New (3)
Unread (11)
Top
New Topic
Announcements
This category is for announcements of Swift evolution proposal reviews and results, as well as other...
Pitches
The Pitches category is an area for pitching ideas for evolution of the Swift language prior to a formal review.
Proposal Reviews
This category is for posting Swift Evolution proposals for review and feedback.
Discussion
The Evolution Discussion category is for general discussion of the evolution of the Swift language.
Topic
Replies
Views
Activity
Introducing Sequence.last?
Pitches
6
65
1h
RFC: making Swift.org a more valuable resource for the Swift community
23
Discussion
50
7.2k
16h
Adding isPowerOf2 to BinaryInteger
Pitches
64
2.3k
17h
[Pre-pitch] removeAll(at:)
Discussion
collection
2
221
18h
Remove duplicate elements from a collection
Pitches
collection
pitch
28
1.1k
20h
Implement buffer pointer's `MutableCollection` methods as nonmutating
Discussion
3
144
22h


Swift
The Swift Forums are governed by the Swift Code of Conduct
Evolution
all
all tags
Latest
New (3)
Unread (11)
Top
New Topic
Announcements
This category is for announcements of Swift evolution proposal reviews and results, as well as other...
Pitches
The Pitches category is an area for pitching ideas for evolution of the Swift language prior to a formal review.
Proposal Reviews
This category is for posting Swift Evolution proposals for review and feedback.
Discussion
The Evolution Discussion category is for general discussion of the evolution of the Swift language.
forums.swift.org/c/evolution
Topic
Views
Activity
Introducing Sequence.last?
Pitches
6
65
1h
RFC: making Swift.org a more valuable resource for the Swift community
23
Discussion
50
7.2k
16h
Adding isPowerOf2 to BinaryInteger
Pitches
64
2.3k
17h
[Pre-pitch] removeAll(at:)
Discussion
collection
2
221
18h
Remove duplicate elements from a collection
Pitches
collection
pitch
28
1.1k
20h
Implement buffer pointer's `MutableCollection` methods as nonmutating
Discussion
3
144
22h

Make Your Swift APIs Better

Expressive

Clear

Easy to use

Make Your Swift APIs Better

Expressive

Clear

Easy to use

Modern Swift API Design

Thursday, 2:00PM

More Information

developer.apple.com/wwdc19/402

Creating Swift Packages	Thursday, 9:00AM
Modern Swift API Design	Thursday 2:00PM
Binary Frameworks in Swift	Thursday, 3:00PM


WWDC19