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
Swift Adoption at Apple
Swift Adoption at Apple
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
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.

<table>
<thead>
<tr>
<th>Download</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xcode (Debugging Symbols)</td>
<td>May 31, 2016</td>
</tr>
<tr>
<td>Ubuntu 15.10 (Signature)</td>
<td>May 31, 2016</td>
</tr>
<tr>
<td>Ubuntu 14.04 (Signature)</td>
<td>May 31, 2016</td>
</tr>
</tbody>
</table>
GitHub
Swift.org Projects on GitHub
Swift.org Projects on GitHub

- swift
- swift-llbuild
- swift-lldb
- swift-llvm
- swift-corelibs-xctest
- swift-package-manager
- swift-evolution
- 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

Package Manager

swift-package-manager
swift-llbuild

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
Foundation on Linux
Foundation on Linux

- URLQueryItem
- AffineTransform
- CharacterSet
- URL
- IndexPath
- Date
- DateInterval
- IndexSet
- Data
- UUID
- Notification
- Measurement
- URLRequest
- PersonNameComponents
- URLComponents
- Decimal
- DateComponents
Language

swift
swift-evolution
Language

swift

swift-evolution
swift-evolution
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

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:

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

declares `foo(_:)` and not `foo(_:__,__):`.
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
<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001-keywords-as-argument-labels.md</td>
<td>SE-0001 &quot;Allow (most) keywords as argument labels&quot; is now implemented.</td>
<td>5 months ago</td>
</tr>
<tr>
<td>0002-remove-currying.md</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0003-remove-var-parameters.md</td>
<td>Update 0003-remove-var-parameters.md (#292)</td>
<td>23 days ago</td>
</tr>
<tr>
<td>0004-remove-pre-post-inc-decrement.md</td>
<td>Doug doesn't want to speculate about the direction of numerics.</td>
<td>6 months ago</td>
</tr>
<tr>
<td>0005-objective-c-name-translation.md</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0006-apply-api-guidelines-to-the-stand...</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0007-remove-c-style-for-loops.md</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0008-lazy-flatmap-for-optionalss.md</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0009-require-self-for-accessing-instan...</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0010-add-escaping-string-unicodescalarvi...</td>
<td>Added review thread link for SE-0010</td>
<td>2 months ago</td>
</tr>
<tr>
<td>0011-replace-typealias-associated.md</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0012-add-noescape-to-public-library-a...</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0013-remove-partial-application-super...</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0014-constrained-AnySequence.md</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0015-tuple-comparison-operators.md</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
<tr>
<td>0016-initializers-for-converting-unsafe...</td>
<td>Bring SE-0016's revision history in line with other proposals. (#296)</td>
<td>24 days ago</td>
</tr>
<tr>
<td>0017-convert-unmanaged-to-use-unsa...</td>
<td>several newly accepted proposals.</td>
<td>23 days ago</td>
</tr>
<tr>
<td>0018-flexible-member-access-initializer.md</td>
<td>Stretch a pedantic itch: change &quot;Author(s)&quot; to either &quot;Author&quot; or &quot;Au...</td>
<td>25 days ago</td>
</tr>
</tbody>
</table>
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
Carefully studied what is important in API design

• Strive for clarity—not terseness or verbosity
• Capture essential information
• Omit redundant information/boilerplate
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/
Example API Changes

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

Swift.Array

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

Foundation.NSURL

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

Swift.Array

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

Foundation.NSURL

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

Swift.Array

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.URLByAppendingPathComponent("file.txt")
```

Foundation.NSURL
Example API Changes

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

Swift.Array

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

Foundation.NSURL
```
Example API Changes

Swift.Array

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

Foundation.NSURL

```swift
if url.isFileURL {}
x = url.appendingPathComponent("file.txt")
```
Importing Objective-C APIs
void CGContextFillPath(CGContextRef);
void CGContextFillPath(CGContextRef);

func CGContextFillPath(_: CGContext)

Swift 2
void CGContextFillPath(CGContextRef)
NS_SWIFT_NAME(CGContext.fillPath(self:));

func CGContextFillPath(_: CGContext)
extension CGContext {
  func fillPath()
}

Swift 2
Swift 3

Swift API Design Guidelines
Presidio
Tuesday 10:00AM

SE-0044
func findAnimals() {
    let request = NSFetchRequest(entityName:"Animal")
    guard let searchResults = 
        try? context.executeFetchRequest(request) as! [Animal] {
        return
    } 
    ... 
    use(searchResults)
}
func findAnimals() {
    let request : NSFetchRequest<Animal> = Animal.fetchRequest
    guard let searchResults = try? context.fetch(request) {
        return
    }
    ...
    use(searchResults)
}
Stringly Typed Objective-C Constants

typedef NSString *NSNotificationName;
const NSNotificationName NSUserDefaultsDidChangeNotification;
Stringly Typed Objective-C Constants

typedef NSString *NSNotificationName;
const NSNotificationName NSUserDefaultsDidChangeNotification;

Imported definition
let NSUserDefaultsDidChangeNotification: String
Stringly Typed Objective-C Constants

typedef NSString *NSNotificationName;
const NSNotificationName NSUserDefaultsDidChangeNotification;

Imported definition

let NSUserDefaultsDidChangeNotification: String

center.addObserver(forName: NSUserDefaultsDidChangeNotification, ...)
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, ...)
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, ...)
Stringly Typed Objective-C Constants

typedef NSString *NSNotificationName NS_EXTENSIBLE_STRING_ENUM;
const NSNotificationName NSUserDefaultsDidChangeNotification;

Imported definition

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

Use

```objective-c
center.addObserver(forName: NSUserDefaultsDidChangeNotification, ...)
```
Stringly Typed Objective-C Constants

typedef NSString *NSNotificationName NS_EXTENSIBLE_STRING_ENUM;
const NSNotificationName NSUserDefaultsDidChangeNotification;

Imported definition

```swift
extension UserDefaults {
    class letDidChangeNotification: NSNotification.Name
}
```

Use

```swift
center.addObserver(forName: UserDefaults.didChangeNotification, ...)```
Strongly Typed Objective-C Constants

typedef NSString *NSNotificationName NS_EXTENSIBLE_STRING_ENUM;
const NSNotificationNameNSUserDefaultsDidChangeNotification;

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

Concurrent Programming with GCD in Swift 3
Core Language
Consistent Parameter Labeling

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

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

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

Swift 2
Consistent Parameter Labeling

```swift
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 `_`.

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

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

**Swift 3**

SE-0046
func anyCommon<T: Sequence, U: Sequence>(lhs: T, rhs: U) -> Bool {

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

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
    >(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
Warn on Unused Results by Default

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

plusOne(x)
Warn on Unused Results by Default

```swift
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

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

```
plusOne(x)
_ = plusOne(x)
```

Result of call to ‘plusOne’ is unused

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)
```
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?

<table>
<thead>
<tr>
<th>SE-0002</th>
<th>Currying func declaration syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE-0003</td>
<td><code>var</code> in function parameter lists</td>
</tr>
<tr>
<td>SE-0004</td>
<td><code>++</code> and <code>--</code> operators</td>
</tr>
<tr>
<td>SE-0007</td>
<td>C-style <code>for</code> loop</td>
</tr>
<tr>
<td>SE-0029</td>
<td>Implicit tuple splat in calls</td>
</tr>
</tbody>
</table>
Core Language
Other small enhancements
Core Language

Other small enhancements
### Core Language

#### Other small enhancements

| 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
| 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

Goal
• Simpler, more consistent, and more predictable type system
• Remove “gotchas” and surprising behavior
• Improve type checker performance
UnsafePointer Nullability

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

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

Swift 2
let ptr : UnsafeMutablePointer<Int>? = nil

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

Swift 3
UnsafePointer Nullability

let ptr : UnsafeMutablePointer<Int>? = nil

ptr .memory = 42

Swift 3

Imported C pointers in APIs obey _Nullable and _Null_unspecifed
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)

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

Swift 2

SE-0054
func f(value: Int!) {
    let x = value + 1
    let y = value
}

Swift 2
func f(value : Int!) {
    let x = value + 1  // x: Int - force unwrapped
    let y = value      // y: Int!
}

Swift 2
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)

```swift
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
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)

```swift
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

\[
i = \text{collection.startIndex}
\]

\[
\text{next} = i.\text{successor}()
\]

Swift 2
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

Swift 2

```swift
i = collection.startIndex
next = i.successor()
```

Swift 3

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

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
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)
```

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

**Swift 2**

**Swift 3**
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
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

<table>
<thead>
<tr>
<th>SE-0008</th>
<th>Add a Lazy <code>flatMap</code> for sequences of optionals</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE-0016</td>
<td>Conversions <code>Unsafe[Mutable]Pointer</code> to <code>Int</code> and <code>UInt</code></td>
</tr>
<tr>
<td>SE-0017</td>
<td>Change <code>Unmanaged</code> to use <code>UnsafePointer</code></td>
</tr>
<tr>
<td>SE-0032</td>
<td>Add <code>first(where:)</code> method to <code>Sequence</code></td>
</tr>
<tr>
<td>SE-0061</td>
<td>Add generic result and error handling to <code>autoreleasepool()</code></td>
</tr>
<tr>
<td>SE-0080</td>
<td>Failable numeric conversion initializers</td>
</tr>
<tr>
<td>SE-0093</td>
<td>Adding a public <code>base</code> property to slices</td>
</tr>
<tr>
<td>SE-0094</td>
<td>Add <code>sequence(first:next:)</code> and <code>sequence(state:next:)</code> to the stdlib</td>
</tr>
</tbody>
</table>
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
Faster Startup Time
Smaller Binaries
String Hashing
Faster Dictionary
Incremental Compilation
Stack Promotion
Dictionary<String, T>
Heap to Stack Promotion for Classes
String Algorithm Optimizations

86x
Whole Module Optimization
Whole Module Optimization

File1.swift → Compiler → File1.o
File2.swift → Compiler → File2.o
File3.swift → Compiler → File3.o
Whole Module Optimization

File1.swift -> Compiler -> File1.o

File2.swift -> Compiler -> File2.o

File3.swift -> Compiler -> File3.o
Whole Module Optimization

File1.swift

File2.swift

Compiler

File2.o

File3.swift
WMO on by Default for New Projects
What About Compile Time?

Files:
- File1.swift
- File2.swift
- File3.swift

Compiler

Output:
- Module.o
What About Compile Time?

File1.swift

File2.swift

File3.swift

Compiler

Module.o
What About Compile Time?

File1.swift

File2.swift

File3.swift

Compiler

Module.o
Code Size Optimization

- DemoBots: 77%
- AlamoFire: 95%

Code Size—Swift 3 vs. Swift 2.2
Synthesized Interfaces

//: Playground – noun: a place where people can play

let exampleArray = Array<Int>()
exampleArray.sort()
Synthesized Interfaces

```swift
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

```swift
extension Array where Element : Comparable {
    /// Returns the elements of the collection, sorted.
    /// ...
    public func sorted() -> [Element]
    /// ...
}
```
Structure

**Array**

An ordered, random-access collection.

---

**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:

```swift
// 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:

```swift
// 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.
Migrating from Swift 2.2
Migrating from Swift 2.2

Choose Swift version:

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

    let frame = UIGraphicsGetImageFromCurrentImageContext()
    UIGraphicsEndImageContext()

    return frame
}
Working with Swift 2.2 and Swift 2.3

```swift
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

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

    let frame = UIGraphicsGetImageFromCurrentImageContext()
    UIGraphicsEndImageContext()

    #if swift(>=2.3)
        return frame!
    #else
        return frame
    #endif
}
```
From Swift 2.3 to Swift 3
Summary

Swift 3 focuses on fundamentals
See swift.org for how to get involved
Migrator available to Swift 3
More Information

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swift API Design Guidelines</td>
<td>Presidio</td>
<td>Tuesday 10:00AM</td>
</tr>
<tr>
<td>Getting Started with Swift</td>
<td>Pacific Heights</td>
<td>Tuesday 1:40PM</td>
</tr>
<tr>
<td>What’s New in Foundation for Swift</td>
<td>Mission</td>
<td>Tuesday 4:00PM</td>
</tr>
<tr>
<td>Introducing Swift Playgrounds</td>
<td>Mission</td>
<td>Wednesday 11:00AM</td>
</tr>
<tr>
<td>Going Server-Side with Swift Open Source</td>
<td>Mission</td>
<td>Friday 9:00AM</td>
</tr>
<tr>
<td>Understanding Swift Performance</td>
<td>Mission</td>
<td>Friday 11:00AM</td>
</tr>
<tr>
<td>Concurrent Programming with GCD in Swift 3</td>
<td>Pacific Heights</td>
<td>Friday 4:00PM</td>
</tr>
<tr>
<td>Swift Open Hours</td>
<td>Developer Tools Lab A</td>
<td>Date/Time</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tuesday 12:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tuesday 3:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wednesday 12:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wednesday 3:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Thursday 12:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Thursday 3:00PM</td>
</tr>
<tr>
<td>Swift Open Hours</td>
<td>Developer Tools Lab A</td>
<td>Date and Time</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wednesday 12:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wednesday 3:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Thursday 12:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Thursday 3:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Friday 9:00AM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Friday 12:00PM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Friday 3:00PM</td>
</tr>
</tbody>
</table>