Make our Swift better

09/05/2018 try! Swift NYC

Daiki Matsudate / @d_date

folio

Daiki Matsudate

@d_date

iOS / macOS Application Development

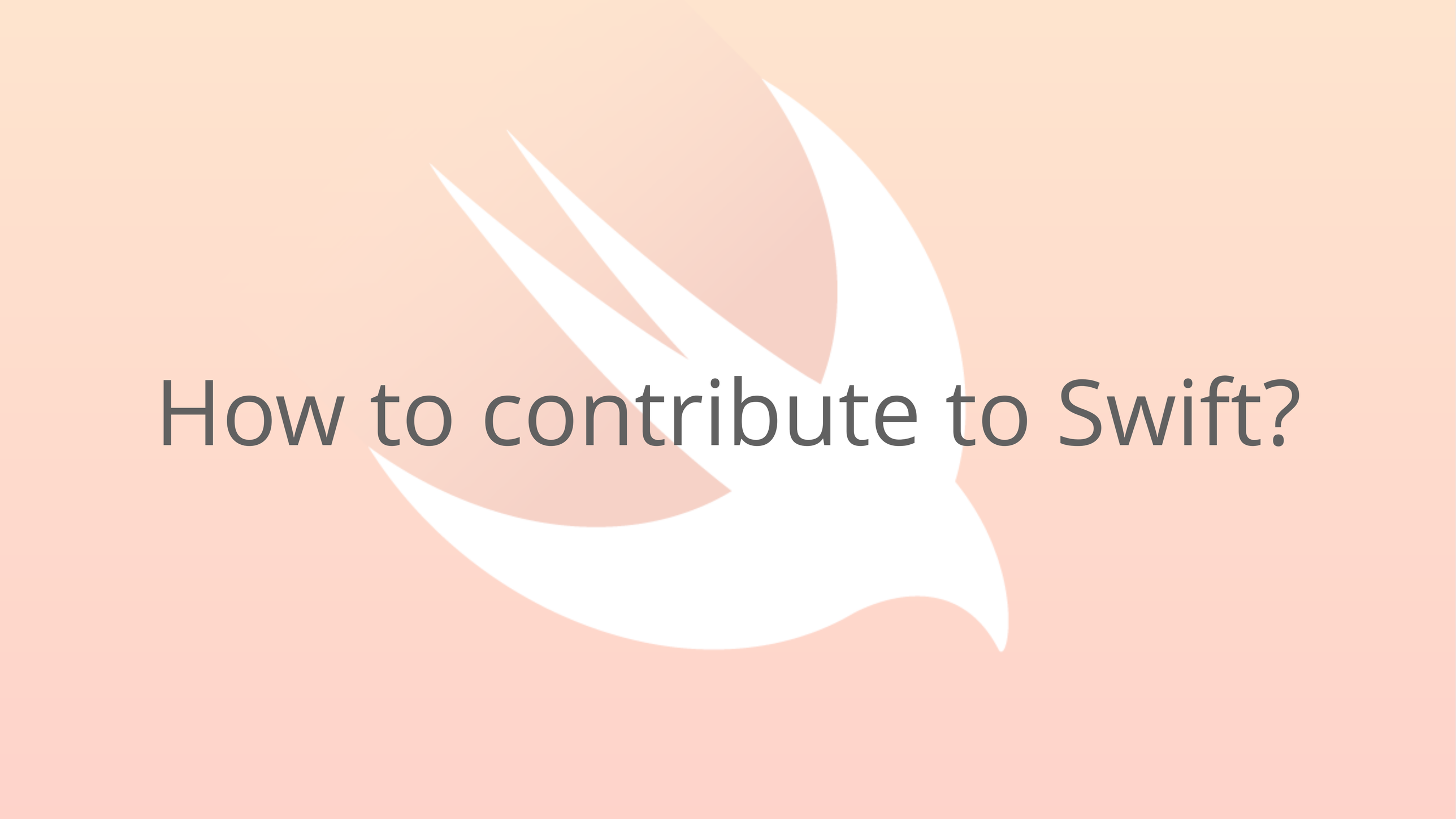
iOS / macOS Application Development

Swift

- First released on June 4, 2014 (Latest 4.2 dev)
- Open Source from Swift 3

Open Source

- You can see whole source code
- You can open issues on repo
- You can make pull request for repo


How to contribute to Swift?

```
static func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
    var destination = [String: String]()
    for (key, nillableValue) in source {
        if let value: Any = nillableValue {
            destination[key] = "\(value)"
        }
    }
    return destination
}
```

```
static func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
    return source.flatMap { $0 }

}
```

```swift
static func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
    return source.flatMap { $0 }
    // [(key: String, value: String)]
}
```

0. Get advice from community💡

Discord: Swift-developers-Japan

DISCORD

d_date Last Wednesday at 4:49 PM

I want it to be more cleaner. How do we that?

```
static func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
        var destination = [String: String]()
        for (key, nillableValue) in source {
            if let value: Any = nillableValue {
                destination[key] = "\(value)"
            }
        }
        return destination
    }
```

tarunon Last Wednesday at 4:57 PM

By using inout reduce, it might be cleaner I think.

tarunon Last Wednesday at 5:00 PM

```
func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
    return source.reduce(into: [String: String](), { (result, x) in
        if let value = x.value else { result[x.key] = "\(value)" }
    })
}
```

(edited)

```
static func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
    var destination = [String: String]()
    for (key, nillableValue) in source {
        if let value: Any = nillableValue {
            destination[key] = "\(value)"
        }
    }
    return destination
}
```

```
static func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
    return source.reduce(into: [String: String](), { (result, x) in
        if let value = x.value { result[x.key] = "\(value)" }
    })
}
```

```
static func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
    return source.reduce(into: [String: String](), { (result, x) in
        if let value = x.value { result[x.key] = "\(value)" }
    })
}

public func reduce<Result>(into initialResult: Result,
                  _ updateAccumulatingResult: (inout Result,
                     (key: Key, value: Value)) throws -> ()) rethrows -> Result
```

```swift
static func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
    return source.reduce(into: [String: String](), { (result, x) in
        if let value = x.value { result[x.key] = "\(value)" }
    })
}

rejectNilHeaders(["a": "1", "b": nil, "c": "3"])
// ["a" : 1, "c" : 3]
```

```swift
static func rejectNilHeaders(_ source: [String: Any?]) -> [String: String] {
    return source.reduce(into: [String: String](), { (result, x) in
        if let value = x.value { result[x.key] = "\(value)" }
    })
}
```

```swift
["a": "1", "b": nil, "c": “3”].xxx({ $0 })
// ["a" : 1, "c" : 3]
```

map and flatMap in Collection

```swift
let r = [1, 2, 3].map { $0 * 2 }
// [2, 4, 6]
```

```swift
let r22 = [1, nil, 3].flatMap { $0 }
// [1, 3]
```

map and compactMap in Collection

```swift
let r = [1, 2, 3].map { $0 * 2 }
// [2, 4, 6]
```

```swift
let r22 = [1, nil, 3].compactMap { $0 }
// [1, 3]
```

mapValues in Dictionary

```
["a": 1, "b": 2, "c": 3].mapValues({ $0 * 2 })
// ["a": 2, "b": 4, "c": 6]
```

map

compactMap

mapValues

map

compactMap

mapValues

?

Dictionary.compactMapValues

map

compactMap

mapValues

compactMapValues

```
["a": "1", "b": nil, "c": “3”].compactMapValues({ $0 })
// ["a" : 1, "c" : 3]
```

```
["a": "1", "b": nil, "c": “3”].compactMapValues({ $0 })
// ["a" : 1, "c" : 3]
```

```
extension Dictionary {
    public func compactMapValues<T>(_ transform: (Value) throws -> T?) rethrows -> [Key: T] {
        return try self.reduce(into: [Key: T](), { (result, x) in
            if let value = try transform(x.value) {
                result[x.key] = value
            }
        })
    }
}
```

1.Post your idea to Forum

https://forums.swift.org

Swift

The Swift Forums are governed by the Swift Code of Conduct

Swift CI will be down for a window of 3-4 hours starting at 1 PM on Jan 25 in order to update the CI bots to the new Xcode beta. (https://developer.apple.com/download/)

Evolution | all in Evolution | all tags | Latest | New (6) | Unread (1) | Top | New Topic

Announcements

This category is for announcements of Swift evolution proposal reviews and results, as well as other administrative...

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	Category	Users	Replies	Views	Activity
Combining hashes • new	Discussion		0	2	1m
SE-0195 — Introduce User-defined “Dynamic Member Lookup” Types	Proposal Reviews		40	2.3k	21m
SE-0197 — Add in-place remove(where:) • new	Proposal Reviews		21	560	3h
[Concurrency] async/await + actors	Discussion		44	71	3h
Let Optional, Dictionary and Array conditionally conform to Hashable • new	Pitches		2	94	4h

[Pitch] Add compactMapValues to Dictionary

Evolution Pitches

d-date Daiki Matsudate 1 3d

Hi,

I'd like to propose about `compactMapValues` in Dictionary.

When I imagine removing nil value from dictionary at first as below, but it doesn't works.

```
["1" :  "1", "2" : nil, "3" :  "3"].flatMap { $0 }

// [(key: "2", value: nil), (key: "1", value: Optional("1")), (key: "3", value: O
```

To avoid this, I've heard using `reduce` like this.

```
let result = ["1": "1", "2": nil, "3": "3"].reduce(into: [String: String]()) { (r
    if let value = x.value { result[x.key] = value }
}

// ["1": "1", "3": "3"]
```

However now that we have `Dictionary.mapValues()` from Swift 4 and the `Sequence.flatMap()` will be renamed to `Sequence.compactMap()` with [SE-0187] 2 . I want `compactMapValues` to remove nil value in dictionary as below.

```
["1" :  "1", "2" : nil, "3" : "3"].compactMapValues({ $0 })

// ["1": "1", "3": "3"]
```

Jan 24

1 / 10
Jan 25

1m ago

lorentey Karoy Lorentey

akashivskyy Jan 24

I can see how this would be useful; it seems like a reasonable addition to complete the API. This is a different operation than `mapValues(_:)` followed by `filter(_:)`.

Do you have a specific implementation in mind? (I suspect it might be possible to use `Dictionary` internals to eliminate rehashing in some cases. But it's probably better to just build a brand new `Dictionary` from scratch, like you do with `reduce`.)

2 Replies 1 Reply

moiseev Max Moiseev Jan 24

I can think of 2 implementations. 1) using `reduce(into:_:)` as posted above; and 2) `Dictionary(uniqueKeysWithValues: srcDict.lazy.filter { $0.1 != nil })`. They are not-really-quite-straightforward, perhaps, but neither benefits from being in the standard library from the performance standpoint. So I'm in the fence about including it.

Upd: and just to contradict myself, the `.lazy.filter` variant is incorrect as it does not unwrap the optionals. So, one extra point to being included.

1 Reply 2 Reply

stephencelis 1 1d

>  lorentey:
>
> I think we should have more practical examples to motivate inclusion. Dictionaries of optional values do not occur very often in my experience – I'd typically just declare a non-optional value and filter out nils during insertion. For other cases, a filter followed by mapValues would often work just as well as compactMapValues.

We added a `compact` method to `Dictionary` when I was at Kickstarter and used it throughout:

- https://github.com/kickstarter/ios-oss/blob/996ba3d8990c85ac9b47dc15e6362a23af49ed69/KsApi/lib/Route.swift#L119-L123
- https://github.com/kickstarter/ios-oss/blob/996ba3d8990c85ac9b47dc15e6362a23af49ed69/KsApi/lib/Route.swift#L308-L312
- https://github.com/kickstarter/ios-oss/blob/a7636fa76c5c97bc87882ca17212be8e944c795e/Kickstarter-iOS/ViewModels/RootViewModel.swift#L108-L116
- https://github.com/kickstarter/ios-oss/blob/1b21ce9100edc2700b30f41432f4c6077febba69/Kickstarter-iOS/ViewModels/AppDelegateViewModel.swift#L942

I even recently added a `filterMapValues` (and `filteredValues`) to our code base at Point-Free:

- https://github.com/pointfreeco/pointfreeco/blob/f97a9e4830a5bdfe778daca8ed771f7fa9eadbc7/Sources/PointFree/Stripe.swift#L306-L311

We try to avoid mutation and write code that's declarative, so our uses tend to be where we want to pass a dictionary literal to a call site that expects non-optional values, but where some of our inputs may be optional.

These use cases may be minor enough to not require an implementation in the standard library, but we'll probably continue to add our own extension and benefit from it regardless. It's a bit messy to `filter` and `mapValues` everywhere (especially with the force unwrap). What we want is succinctness and safety.

nnnnnnnn Nate Cook

lorentey Jan 25

This isn't quite so much about dictionaries with optional values as transformations of a dictionary's values that yield an optional. For example:

```
let d = ["a": "1", "b": "2", "c": "three"]
// This is .compactMapValues(Int.init):
let e = d.mapValues(Int.init).filter({ $0.value != nil }).mapValues({ $0! })
```

There's no *real* savings here from a hashing perspective—the filtering means we'll need to rehash all the elements—but it would save on allocating more space than we need for a bunch of `nil` values.

5 ❤ Reply

```swift
extension Dictionary {
    public func compactMapValues<T>(_ transform: (Value) throws -> T?) rethrows -> [Key: T] {
        return try self.reduce(into: [Key: T](), { (result, x) in
            if let value = try transform(x.value) {
                result[x.key] = value
            }
        })
    }
}
```

2. Make proposal to Swift-evolution

apple / swift-evolution

Unwatch 1,276 | Unstar 9,103 | Fork 1,460

Code | Pull requests 18 | Projects 0 | Insights

This maintains proposals for changes and user-visible enhancements to the Swift Programming Language.
https://apple.github.io/swift-evolution/

2,460 commits | **8** branches | **0** releases | **216** contributors | Apache-2.0

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

tkremenek Merge pull request #897 from apple/SE-0224-revision ... Latest commit `fd9f3cb` an hour ago

proposal-templates	Add SwiftPM proposal template	2 months ago
proposals	Merge pull request #897 from apple/SE-0224-revision	an hour ago
releases	Update swift-3_0.md	6 months ago
.gitignore	Initial content: Swift evolution process, proposal template, license	3 years ago
.nojekyll	[Status] Disable Jekyll for GitHub Pages	2 years ago
CONTRIBUTING.md	Update the status page to use data from data.swift.org.	2 years ago
LICENSE.txt	Initial content: Swift evolution process, proposal template, license	3 years ago

2. Make proposal to Swift-evolution

103 proposals/0218-introduce-compact-map-values.md

View

```
@@ -0,0 +1,103 @@
+# Introduce `compactMapValues` to Dictionary
+
+* Proposal: [SE-0218](0218-introduce-compact-map-values.md)
+* Authors: [Daiki Matsudate](https://github.com/d-date)
+* Review Manager: [Ben Cohen](https://github.com/airspeedswift)
+* Status: **Pending Review (July 3-9 2018)**
+
+<!-- *During the review process, add the following fields as needed:* -->
+
+* Implementation: [apple/swift#15017](https://github.com/apple/swift/pull/15017)
+* Pitch: [Forum thread](https://forums.swift.org/t/add-compactmapvalues-to-dictionary/8741)
+<!-- * Decision Notes: [Rationale](https://lists.swift.org/pipermail/swift-evolution/), [Additional Commentary]
 (https://lists.swift.org/pipermail/swift-evolution/) -->
+<!-- * Bugs: [SR-NNNN](https://bugs.swift.org/browse/SR-NNNN), [SR-MMMM](https://bugs.swift.org/browse/SR-MMMM) -->
+<!-- * Previous Revision: [1](https://github.com/apple/swift-evolution/blob/...commit-ID.../proposals/NNNN-filename.md)
 -->
+<!-- * Previous Proposal: [SE-XXXX](XXXX-filename.md) -->
+
+## Introduction
+
+This proposal adds a combined filter/map operation to `Dictionary`, as a companion to the `mapValues` and filter
 methods introduced by [SE-0165](https://github.com/apple/swift-evolution/blob/master/proposals/0165-dict.md). The new
 compactMapValues operation corresponds to compactMap on Sequence.
```

Proposal

- Introuduction
- Motivation
- Proposed Solution
- Detailed Design
- Source compatibility
- Effect on ABI stability / API resilience
- Alternative considered

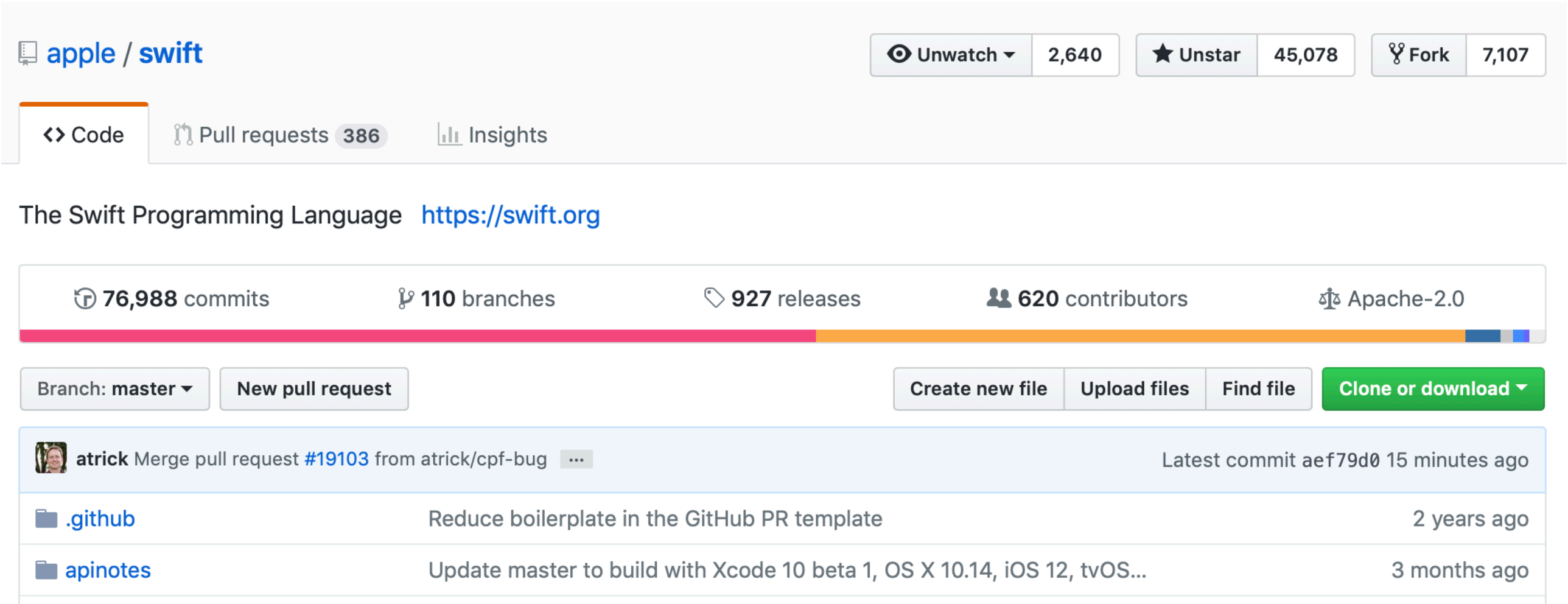
2. Make proposal to Swift-evolution

lorentey approved these changes on Mar 20

3. Build your environment

3. Build your environment

```
$ git clone https://github.com/apple/swift.git
```



3. Build your environment

```
$ ./swift/utils/update-checkout --clone
```

3. Build your environment

```
$ cd swift

$ utils/build-script -Rt
```

3. Build your environment

Waiting for 30 minutes

4. Implement your idea

4. Implement your idea

```
/// Returns a new dictionary containing the keys of this dictionary with the
/// values transformed by the given closure.
/// - Parameter transform: A closure that transforms a value. `transform`
///   accepts each value of the dictionary as its parameter and returns a
///   transformed value of the same or of a different type.
/// - Returns: A dictionary containing the keys and transformed values of
///   this dictionary.
@inlineable // FIXME(sil-serialize-all)
public func compactMapValues<T>(
  _ transform: (Value) throws -> T?
) rethrows -> Dictionary<Key, T> {
  return try self.reduce(into: [Key: T](), { (result, x) in
    if let value = try transform(x.value) {
      result[x.key] = value
    }
  })
}
```

5. Test your implementation

5. Test your implementation

```
// RUN: %target-run-simple-swift
// REQUIRES: executable_test
// REQUIRES: objc_interop

import Foundation
import StdlibUnittest

var tests = TestSuite("CompactMapValues")

tests.test("DefaultReturnType") {
    var result = ["a": "1", "c": "3"].compactMapValues { $0 }
    expectType([String: String].self, &result)
}

tests.test("ExplicitTypeContext") {
    expectEqual(["a":"1","c":"3"],
                ["a":"1","b":nil,"c":"3"].compactMapValues({$0})
    )
    expectEqual(["a": 1, "b": 2],
                ["a":"1","b":"2", "c":"three"].compactMapValues(Int.init)
    )
}

runAllTests()
```

5. Test your implement

```
// RUN: %target-run-simple-swift
// REQUIRES: executable_test
// REQUIRES: objc_interop

import Foundation
import StdlibUnittest

var tests = TestSuite("CompactMapValues")

tests.test("DefaultReturnType") {
    var result = ["a": "1", "c": "3"].compactMapValues { $0 }
    expectType([String: String].self, &result)
}

tests.test("ExplicitTypeContext") {
    expectEqual(["a":"1","c":"3"],
                ["a":"1","b":nil,"c":"3"].compactMapValues({$0})
    )
    expectEqual(["a": 1, "b": 2],
                ["a":"1","b":"2", "c":"three"].compactMapValues(Int.init)
    )
}

runAllTests()
```

Import StblibUnittest

Write your test case

5. Test your implement

```
$ utils/build-script -Rt
```

5. Test your implement

Waiting for 10 – 30 minutes ☕

6. Benchmark your changes (Optional)

6. Benchmark your changes (Optional)

```
public let DictionaryCompactMapValues = [
    BenchmarkInfo(name: "DictionaryCompactMapValuesOfNilValue", runFunction: run_DictionaryCompactMapValuesOfNilValue, tags:
[.validation, .api, .Dictionary]),]

@inline(never)
public func run_DictionaryCompactMapValuesOfNilValue(_ N: Int) {
    let size = 100
    var dict = [Int: Int?](minimumCapacity: size)

    // Fill Dictionary
    for i in 1...size {
        if i % 2 == 0 {
            dict[i] = nil
        } else {
            dict[i] = i
        }
    }
    CheckResults(dict.count == size / 2)

    var refDict = [Int: Int]()
    for i in stride(from: 1, to: 100, by: 2) {
        refDict[i] = i
    }

    var newDict = [Int: Int]()
    for _ in 1...1000*N {
        newDict = dict.compactMapValues({$0})
        if newDict != refDict {
            break
        }
    }

    CheckResults(newDict == refDict)
}
```

6. Benchmark your changes (Optional)

```
public let DictionaryCompactMapValues = [
    BenchmarkInfo(name: "DictionaryCompactMapValuesOfNilValue", runFunction: run_DictionaryCompactMapValuesOfNilValue, tags:
[.validation, .api, .Dictionary]),]

@inline(never)
public func run_DictionaryCompactMapValuesOfNilValue(_ N: Int) {
    let size = 100
    var dict = [Int: Int?](minimumCapacity: size)

    // Fill Dictionary
    for i in 1...size {
        if i % 2 == 0 {
            dict[i] = nil
        } else {
            dict[i] = i
        }
    }
    CheckResults(dict.count == size / 2)

    var refDict = [Int: Int]()
    for i in stride(from: 1, to: 100, by: 2) {
        refDict[i] = i
    }

    var newDict = [Int: Int]()
    for _ in 1...1000*N {
        newDict = dict.compactMapValues({$0})
        if newDict != refDict {
            break
        }
    }

    CheckResults(newDict == refDict)
}
```

Specify your benchmark info

Prepare the data

Check compactMapValues work well

6. Benchmark your changes (Optional)

`benchmark/utils/main.swift`

```
import DictionaryCompactMapValues

registerBenchmark(DictionaryCompactMapValues)
```

`benchmark/CMakeLists.txt`

```
single-source/DictionaryCompactMapValues
```

6. Benchmark your changes (Optional)

```
$ swift/utils/build-script --benchmark
```

6. Benchmark your changes (Optional)

Waiting for 2 hours 😴

Improved benchmarking for pull requests

Development Compiler benchmarks

Erik_Eckstein 11d

I'd like to share some exciting news about benchmarking:
We made some significant improvements for running the benchmarks in pull requests:

- It's now a lot faster: down to 30min from 2h (including the compiler build time)
- Reduced noise: almost no false alarms anymore
- Code size differences are now reported - for the benchmark object files and also for the Swift standard library files
- Some improvements of the report table format. For example, improvements are not folded by default but shown in the same table as regressiosn (we should be proud of improvements and not hide them!)

Currently the new feature can be tested with "`@swift-ci` smoke benchmark staging" and they will go live with "`@swift-ci` smoke benchmark" soon.
You can look at a test PR to see some sample output: https://github.com/apple/swift/pull/18876 48

Now what about the non-smoke "`@swift-ci` benchmark"? Currently the only difference between smoke and non-smoke are the number of iterations. But as the new method reduces noise anyway, I'm actually thinking of making "smoke benchmark" the default, i.e. just having "`@swift-ci` benchmark" which does the new thing.

Running Swift-ci on GitHub

lorentey commented on May 15 — Member

@swift-ci please smoke benchmark

swift-ci commented on May 15 — Contributor

Build comment file:

Optimized (O)

▼ Regression (15)

TEST	OLD	NEW	DELTA	SPEEDUP
StringBuilderWithLongSubstring	1465	1986	+35.6%	**0.74x**
DictionarySubscriptDefaultMutationArray	619	693	+12.0%	**0.89x (?)**
MapReduceLazyCollectionShort	34	38	+11.8%	**0.89x**
SubstringComparable	26	29	+11.5%	**0.90x**
SuffixAnyCollectionLazy	20692	22871	+10.5%	**0.90x (?)**
SubstringFromLongString	10	11	+10.0%	**0.91x**
MapReduceAnyCollection	404	442	+9.4%	**0.91x**
StringFromLongWholeSubstringGeneric	22	24	+9.1%	**0.92x**

7. Waiting to starting evolution process

7. Waiting to starting evolution process

SE-0218 — Introduce compactMapValues to Dictionary

Evolution Proposal Reviews

Ben_Cohen 9d

The review of SE-0218 — Introduce `compactMapValues` to `Dictionary` 181 begins now and runs through July 11, 2018.

Reviews are an important part of the Swift evolution process. All review feedback should be either on this forum thread or, if you would like to keep your feedback private, directly to the review manager `@Ben_Cohen` (via email or direct message in the Swift forums).

What goes into a review of a proposal?

The goal of the review process is to improve the proposal under review through constructive criticism and, eventually, determine the direction of Swift.

When reviewing a proposal, here are some questions to consider:

- What is your evaluation of the proposal?

CTMacUser Daryle Walker Jul 5

+1

1 ♥ Reply

Erica_Sadun Jul 5

`compactMapValues` seems like a natural fit for Swift. It provides measurable utility for conversion from `[T: U?]` -> `[T: U]` and for selecting non-nil `[T: V]` entries from `[T: f(U) -> V?]`. It is one of those useful utilities that I've written several times and would love to see be built-in.

I like the current push to include useful features like this. It builds on SE-0165, which has been a godsend (Thank you `@nnnnnnnn`) .

Yes, I believe it fits well with the feel and direction of Swift.

2 ♥ Reply

jawbroken Jul 6

Looks good to me. In future I would also like to see a `compactValues()` method on `Dictionary`, and a matching `compact()` on `Array`, to complete the set here.

2 Replies ⌄ 5 ♥ Reply

[Accepted] SE-0218: Introduce compactMapValues to Dictionary

Evolution Announcements

Ben_Cohen 12h

SE-0218: Introduce `compactMapValues` to `Dictionary` 38 has been **accepted**.

Feedback on the proposal was very positive – the main concern being with the name, but this falls naturally out of our existing method names so is consistent, if clunky.

Thank you to @d-date for proposing and to everyone who participated.

6 Reply

SE-0218 — Introduce compactMapValues to Dictionary

Implemented

SE-0218 Introduce compactMapValues to Dictionary

Author: Daiki Matsudate
Review Manager: Ben Cohen

Implemented In: Swift 5
Implementation: swift#15017

Available in Swift 5

Implemented

SE-0218 Introduce compactMapValues to Dictionary

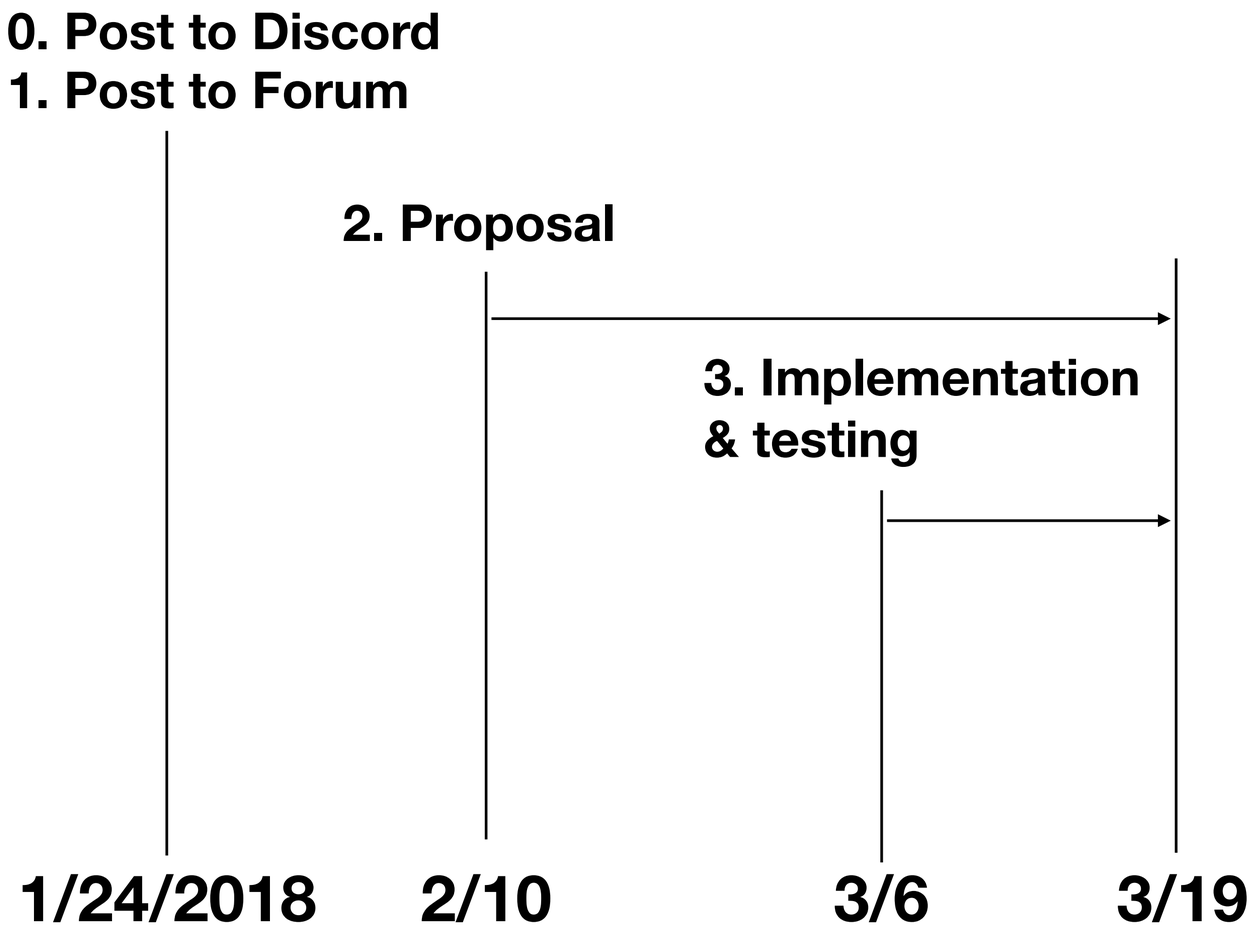
Author: Daiki Matsudate
Review Manager: Ben Cohen

Implemented In: Swift 5
Implementation: swift#15017

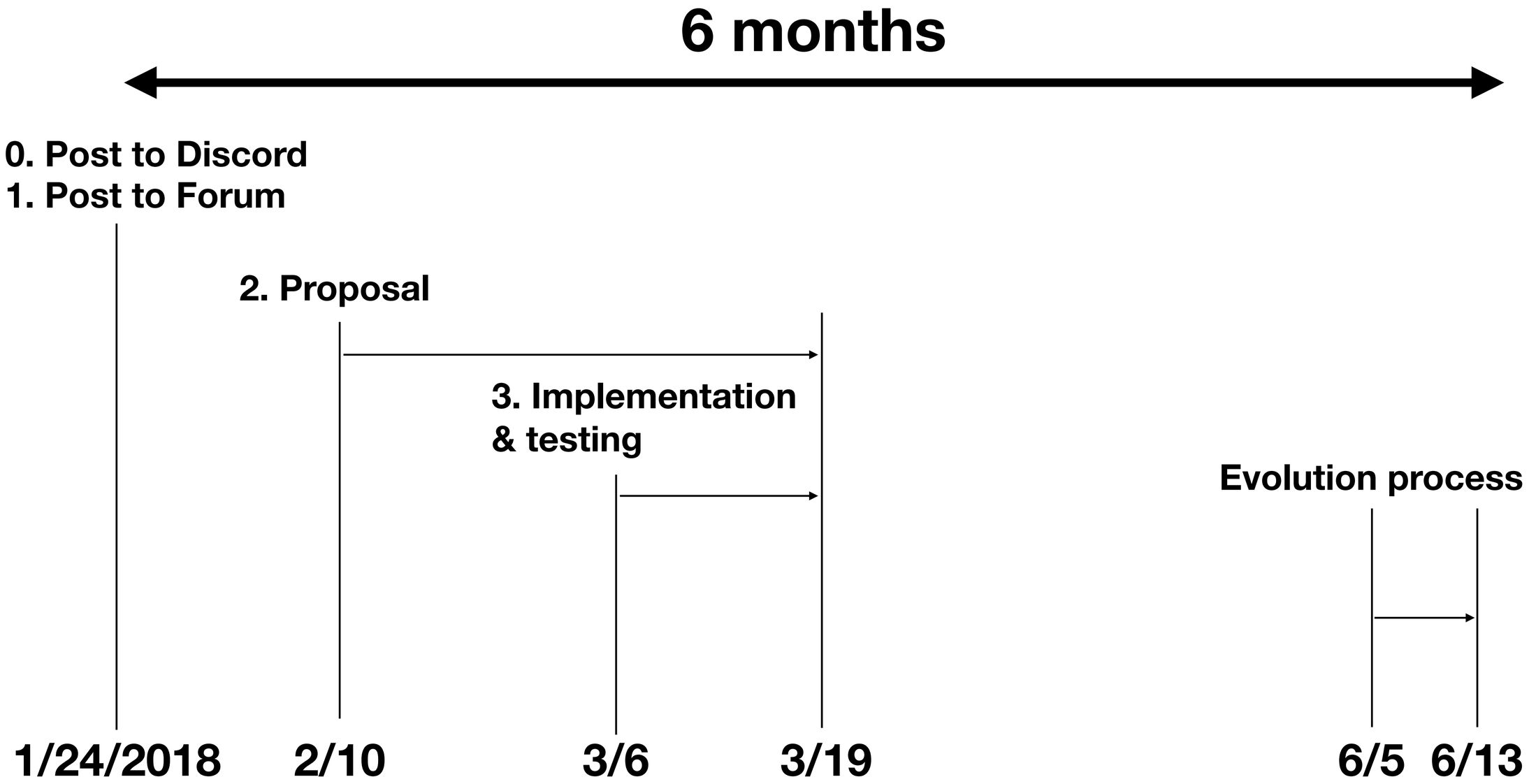
Recap

0. Get advice from local community
1. Post your idea to forum.swift.org
2. Make proposal to Swift-evolution
3. Build your environment
4. Implement your idea
5. Test your implementation
6. Benchmark your changes (Optional)
7. Waiting to starting evolution process

Timeline


0. Post to Discord
1. Post to Forum
2. Proposal
3. Implementation
& testing
1/24/2018
2/10
3/6
3/19

Timeline


6 months
0. Post to Discord
1. Post to Forum
2. Proposal
3. Implementation
& testing
Evolution process
1/24/2018
2/10
3/6
3/19
6/5
6/13

Recap

- Swift is now open source that you can contribute
- Before submitting your idea in Pitch, consider discussed in local community
- The process is open, don’t be shy!

Special Thanks

- Swift-developer-japan
- @tarunon
- try!Swift NYC Organizers / Staffs
- @NatashaTheRobot
- And you!

folio