Getting started with Swift

New programming language


Big Nerd
Ranch
and it's a nice little
incremental improvement
2:04 / 30:10

λ

Functional Swift Conference

December 6th, Brooklyn

On December 6th 2014, a small conference on the topic of Functional Programming in Swift happened in Brooklyn. We had a day filled with with awesome talks and lots of great conversation. Videos of the talks are on the speaker pages.

Speakers

Natasha Murashev
(Video)

Brian Gesiak
(Video)

Justin Spahr-Summers
(Video)

Andy Matuschak
(Video)

John Gallagher
(Video)

Brandon Williams
(Video)

Dec 2015	Dec 2014	Change	Programming Language
1	2	↑	Java
2	1	↓	C
3	4	↑	C++
4	8	⇈	Python
5	5		C#
6	6		PHP
7	9	↑	Visual Basic .NET
8	7	↓	JavaScript
9	10	↑	Perl
10	18	⇈	Ruby
11	32	⇈	Assembly language
12	11	↓	Visual Basic
13	16	↑	Delphi/Object Pascal
14	17	↑	Swift
15	3	⇊	Objective-C

Dec 3, 2015

Swift is Open Source

Swift is now open source. Today Apple launched the open source Swift community, as well as amazing new tools and resources including:

- Swift.org – a site dedicated to the open source Swift community
- Public source code repositories at github.com/apple
- A new Swift package manager project for easily sharing and building code
- A Swift-native core libraries project with higher-level functionality above the standard library
- Platform support for all Apple platforms as well as Linux

Now anyone can download the code and in-development builds to see what the team is up to. More advanced developers interested in contributing to the project can file bugs, participate in the community, and contribute their own fixes and enhancements to make Swift even better. For production App Store development you should always use the stable releases of Swift included in Xcode, and this remains a requirement for app submission.

Sort Descriptors in Swift

From Runtime Magic To Functions

Just last week, someone asked me “in what respect does Swift fall short of the dynamic features of Objective-C”?

Dynamic programming means a lot of different things to different people, and I think they meant runtime programming. In this post, we’ll look at replacing Objective-C’s runtime programming with Swift’s functions.

This post is an excerpt from the Functions chapter in Advanced Swift, which we’re currently rewriting (and making very good progress). The text below was originally written by Airspeed Velocity. I took his text and code, updated everything for Swift 3 and made some heavy edits. Thanks to Ole Begemann for reading through a draft of this.

Unfortunately...

- Standard frameworks are still written in Objective-C, and rely heavily on its runtime
- A lot of development patterns are not *Swifty* at all (delegates, protocols with optional methods, heavy subclassing...)
- `UIViewController`
- Strange runtime errors and crashes

Let's talk language philosophy

Let's talk language philosophy

Swift is friendly to new programmers. It is the first industrial-quality systems programming language that is as expressive and enjoyable as a scripting language.

— The Swift programming language

Familiarity

Language Philosophy: **Familiarity**

```
var numbers = Array(1..<100)
var found42 = false
for number in numbers {
    if number == 42 {
        found42 = true
    }
}
if found42 {
    print("insert 42 meme here")
}
```

Safety

Language Philosophy: **Safety**

```
let text = "42"

let maybeFortyTwo = Int(text) /// Int?

//let eightyFour = maybeFortyTwo * 2 /// ERROR

guard let fortyTwo = maybeFortyTwo else {
    /// do something else
    return
}

let eightyFour = fortyTwo * 2 /// Int
```

Expressiveness

Language Philosophy: **Expressiveness**

```
let toInt: (Character) -> Int = { Int(String($0))! }
let isEven: (Int) -> Bool = { $0 % 2 == 0 }

let sumOfEvens = "0123456789"

    .characters

    .map(toInt)

    .filter(isEven)

    .reduce(0, +)
```

Discoverability

Language Philosophy: **Discoverability**

```
enum HTTPCode: Int {
    case OK = 200
    case notFound = 404
    case internalServerError = 500
}

let receivedCode = HTTPCode(rawValue: 404)!
switch receivedCode {
case .OK: print("good job")
default: print("BUUUUUUUG")
}
```

Language Philosophy: **Discoverability**

```
enum ConnectionResponse {
    case OK(output: Any)
    case notFound(error: String)
    case internalServerError(error: String)
}

let receivedResponse = ConnectionResponse.notFound(error: "There's nothing here")
switch receivedResponse {
case .notFound(let error): print(error)
case .internalServerError(let error): print(error)
default: print("good job")
}
```

Speed

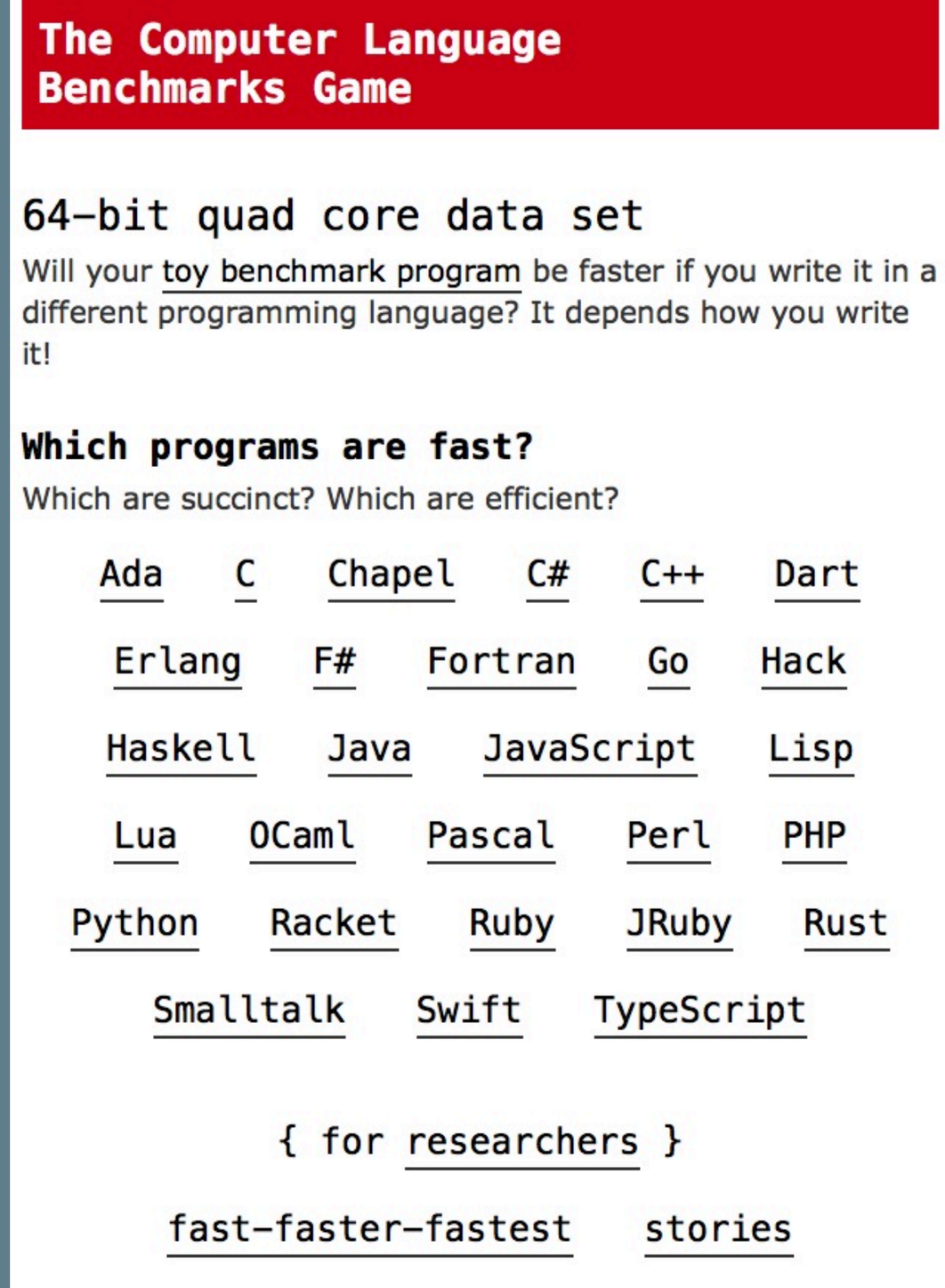
Language Philosophy: **Speed**

Swift is intended as a replacement for C-based languages (C, C++, and Objective-C).

As such, Swift must be comparable to those languages in performance for most tasks.

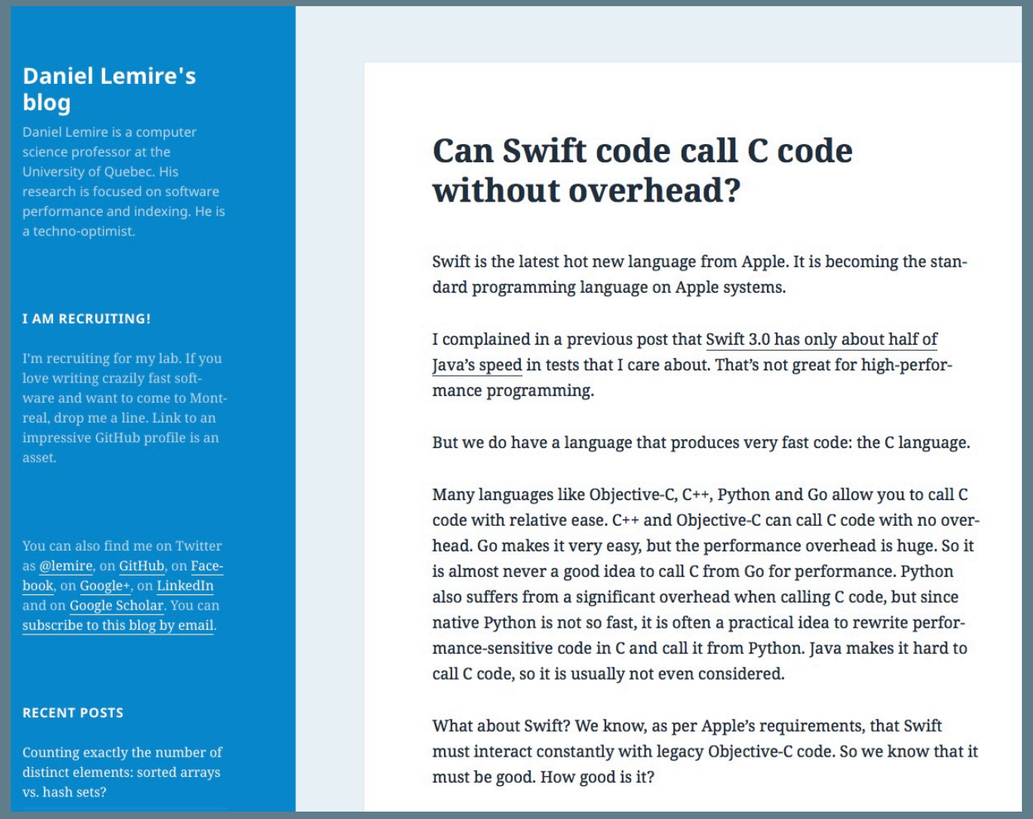
Performance must also be predictable and consistent, not just fast in short bursts that require clean-up later.

There are lots of languages with novel features — being fast is rare.



Language Philosophy: **Speed**

Calling C from Swift has almost zero cost



Daniel Lemire's blog

Daniel Lemire is a computer science professor at the University of Quebec. His research is focused on software performance and indexing. He is a techno-optimist.

I AM RECRUITING!

I'm recruiting for my lab. If you love writing crazily fast software and want to come to Montreal, drop me a line. Link to an impressive GitHub profile is an asset.

You can also find me on Twitter as @lemire, on GitHub, on Facebook, on Google+, on LinkedIn and on Google Scholar. You can subscribe to this blog by email.

RECENT POSTS

Counting exactly the number of distinct elements: sorted arrays vs. hash sets?

Can Swift code call C code without overhead?

Swift is the latest hot new language from Apple. It is becoming the standard programming language on Apple systems.

I complained in a previous post that Swift 3.0 has only about half of Java's speed in tests that I care about. That's not great for high-performance programming.

But we do have a language that produces very fast code: the C language.

Many languages like Objective-C, C++, Python and Go allow you to call C code with relative ease. C++ and Objective-C can call C code with no overhead. Go makes it very easy, but the performance overhead is huge. So it is almost never a good idea to call C from Go for performance. Python also suffers from a significant overhead when calling C code, but since native Python is not so fast, it is often a practical idea to rewrite performance-sensitive code in C and call it from Python. Java makes it hard to call C code, so it is usually not even considered.

What about Swift? We know, as per Apple's requirements, that Swift must interact constantly with legacy Objective-C code. So we know that it must be good. How good is it?

Features

Features: **First-Class Functions**

```
let tableOf3_0 = (1...10).map({ value in
    return value*3
})
```

Features: **First-Class Functions**

```
let tableOf3_1 = (1...10).map { value in
    return value*3
}
```

Features: **First-Class Functions**

```
let tableOf3_2 = (1...10).map { $0*3 }
```

Features: **First-Class Functions**

```
func times3(value: Int) -> Int {
    return value*3
}

let tableOf3_3 = (1...10).map { times3(value: $0) }
```

Features: **First-Class Functions**

```
let tableOf3_4 = (1...10).map(times3)
```

Features: **Value Types**

```
struct Point {
    var x: Int
    var y: Int
}
var current = Point(x: 4, y: 2)
let start = current
current.x *= 3
current.y *= 3
let end = current
let vector = Point(x: end.x - start.x, y: end.y - start.y)
/// Point(x: 8, y: 4)
```

Features: **Value Types**

```
struct Person {
    var firstName: String
    var lastName: String

    func greet(with greeting: String) -> String {
        return "\(greeting) \(firstName) \(lastName)!"
    }
}

let mario = Person(firstName: "Mario", lastName: "Rossi")
print(mario.greet(with: "Da quanto tempo")) /// Da quanto tempo Mario Rossi!
```

Features: **Optionals**

```
let maybeEightyFour = maybeFortyTwo.map { $0*2 }

let maybeTrue: Bool? = true

//if maybeTrue { } /// ERROR

if let x = maybeTrue, x == true {
    print("ok great")
}
```

Features: **Optionals**

```
struct Company { var name: String }

struct Job { var title: String; var company: Company? }

struct Worker {
    var firstName: String
    var lastName: String
    var job: Job?
}
```

Features: **Optionals**

```
let luigi = Worker(
    firstName: "Luigi",
    lastName: "Bianchi",
    job: Job(
        title: "Uomo token",
        company: nil))

let luigisCompanyName = luigi.job?.company?.name /// nil
```

Features: **Initialization**

```
/// error: class 'JobRepository' has no initializers
class JobRepository {
    var jobs: [Job]

    func add(job: Job) {
        jobs.append(job)
    }
}
```

Features: **Initialization**

```
class JobRepository {
    var jobs: [Job]

    func add(job: Job) {
        jobs.append(job)
    }

    init() {
        jobs = []
    }
}
```

Features: **Extensions**

```
extension JobRepository {
    func getJob(withTitle required: String) -> Job? {
        return jobs.first(where: { $0.title == required })
    }
}

let noJob = JobRepository().getJob(withTitle: "Plumber") /// nil
```

Features: **Extensions**

```
extension String {
    var reversed: String {
        return String(characters.reversed())
    }
}
```

Features: **Error Handling**

```
extension String: Error {}

extension JobRepository {
    func getJob(withTitle required: String) throws -> Job {
        if let first = jobs.first(where: { $0.title == required }) {
            return first
        } else {
            throw "No job with title: \(required)"
        }
    }
}

/// error: call can throw but is not marked with 'try'
let noJob = JobRepository().getJob(withTitle: "Plumber")
```

Features: **Error Handling**

```
do {
    let noJob = try JobRepository().getJob(withTitle: "Plumber")
    print("JOB FOUND")
}
catch let error {
    print(error) /// No job with title: Plumber
}
```

Features: **Protocols**

```
class Page {}

enum PresentationStyle {
    case push
    case modal
}

protocol PagePresenter {
    func present(page: Page, style: PresentationStyle)
}
```

Features: **Protocols**

```
class AppNavigator: PagePresenter {
    func present(page: Page, style: PresentationStyle) {
        /// presentation code
    }
}

extension Page {
    func presentModally(with presenter: PagePresenter) {
        presenter.present(page: self, style: .modal)
    }
}
```

Features: **Protocols**

```
extension PagePresenter {
    func presentModally(_ page: Page) {
        present(page: page, style: .modal)
    }
}

protocol PageOwner: PagePresenter {
    var page: Page { get }
}

extension PageOwner {
    func presentModally() {
        present(page: page, style: .modal)
    }
}
```

Features: **Protocols**

```
protocol PagePresenter {
    func present(page: Page, style: PresentationStyle)
}

protocol PageOwner {
    var page: Page { get }
}

extension PageOwner where Self: PagePresenter {
    func presentModally() {
        present(page: page, style: .modal)
    }
}
```

Features: **Generics**

```
struct Resource<A> {
    let name: String
    let parse: (Any) -> A?
}

let id = Resource<Int>(name: "Identifier") {
    guard let value = $0 as? Int, value > 0 else { return nil }
    return value
}
```

Features: **Generics**

```
let possibleIds: [Any] = ["notAnId", 1, 2, 0, -1, "5", 6]

let actualIds = possibleIds.flatMap(id.parse)

print(actualIds) // [1,2,6]
```

Features: **Testing**

```
extension Int {
    func modulo(_ value: Int) -> Int {
        return self % value
    }
}

func testModulo() {
    XCTAssertEqual(3.modulo(2), 1)
}

testModulo()
```

Features: **Testing**

- XCTest is powerful but rudimentary.
- Hard to write *declarative* test cases.
- There are many excellent testing frameworks.
- I like **Quick**, a BDD framework.
- It comes with **Nimble**, a matcher framework that's much nicer to use than XCTest.

Features: **Playgrounds**

- Advanced REPL
- View rendering
- Value history
- Can be added to Xcode projects, to test your code dynamically
- Also on iPad

Features: **Objective-C Interoperability**

- Swift can natively work with Objective-C code and can interact with Objective-C runtime
- Objective-C can only see a *subset* of Swift
- Objective-C is not compatible with *struct* and *enum*
- Swift classes must be subclasses of NSObject
- Either you write Swift *like* Objective-C...
- ...or you write *adapters* that turn Swift code into something visible to Objective-C

KEEP
CALM
AND
USE
OPEN SOURCE

Apple

Cupertino, CA https://developer.apple.com/opensource/

Repositories People 59

Pinned repositories

swift
The Swift Programming Language
C++ ★ 38.6k ⑂ 5.8k

swift-evolution
This maintains proposals for changes and user-visible enhancements to the Swift Programming Language.
JavaScript ★ 7.1k ⑂ 1.1k

Search repositories... Type: All Language: All

swift-lldb

This is the version of LLDB that supports the Swift programming language & REPL.

C++ ★ 404 ⑂ 115 Updated 17 minutes ago

swift-compiler-rt

C ★ 43 ⑂ 28 Updated 17 minutes ago

swift-clang

C++ ★ 451 ⑂ 97 Updated 17 minutes ago

swift-llvm

Top languages

Swift Python C C++ HTML

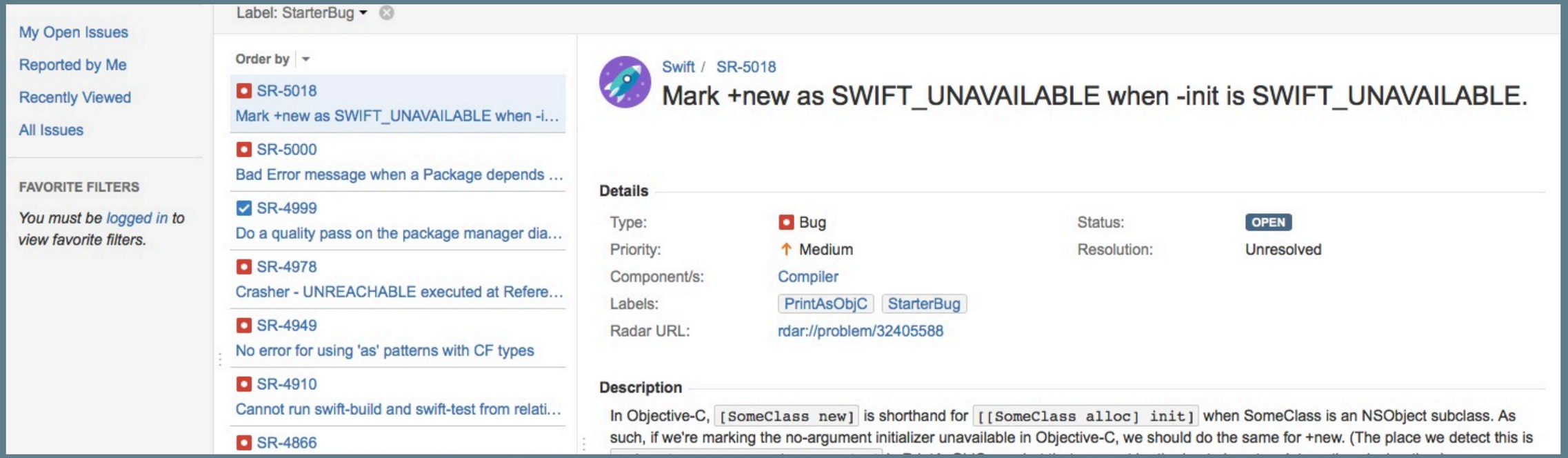
People 59 >

Open Source

- swift.org
- apple/swift
- apple/swift-evolution
- apple/swift-package-manager
- apple/swift-corelibs-foundation
- apple/swift-corelibs-libdispatch

Open Source

- mailing lists
- bugs.swift.org (Jira)



- [swift-evolution] Subclass Existentials *Matthew Johnson*
- [swift-evolution] extending typealiases *Adrian Zubarev*
- [swift-evolution] extending typealiases *Anton Zhilin*
- [swift-evolution] for-else syntax *Jordan Rose*
- [swift-evolution] Compile-time generic specialization *Douglas Gregor*
- [swift-evolution] extending typealiases *Douglas Gregor*
- [swift-evolution] Strings in Swift 4 *David Waite*
- [swift-evolution] extending typealiases *Adrian Zubarev*
- [swift-evolution] Strings in Swift 4 *Ted F.A. van Gaalen*
- [swift-evolution] Warn about unused Optional.some(()) *Jordan Rose*
- [swift-evolution] Warn about unused Optional.some(()) *Daniel Duan*
- [swift-evolution] Subclass Existentials *Karl Wagner*
- [swift-evolution] Warn about unused Optional.some(()) *Jordan Rose*
- [swift-evolution] Warn about unused Optional.some(()) *Daniel Duan*
- [swift-evolution] Warn about unused Optional.some(()) *Jordan Rose*
- [swift-evolution] [swift-build-dev] Package manager support for versioning (both language and tools) *Rick Ballard*
- [swift-evolution] Strings in Swift 4 *David Waite*
- [swift-evolution] Class and Subclass Existentials (Round 2) *David Hart*
- [swift-evolution] define backslash '\' as a operator-head in the swift grammar *Nevin Brackett-Rozinsky*
- [swift-evolution] Class and Subclass Existentials (Round 2) *Adrian Zubarev*
- [swift-evolution] Class and Subclass Existentials (Round 2) *Douglas Gregor*
- [swift-evolution] Strings in Swift 4 *Dave Abrahams*
- [swift-evolution] Removing enumerated? *Dave Abrahams*
- [swift-evolution] [Discussion] mailing list alternative *Dave Abrahams*
- [swift-evolution] Class and Subclass Existentials (Round 2) *Matthew Johnson*
- [swift-evolution] [Discussion] mailing list alternative *Chris Hanson*
- [swift-evolution] Strings in Swift 4 *Ted F.A. van Gaalen*
- [swift-evolution] [Discussion] mailing list alternative *Daniel Duan*

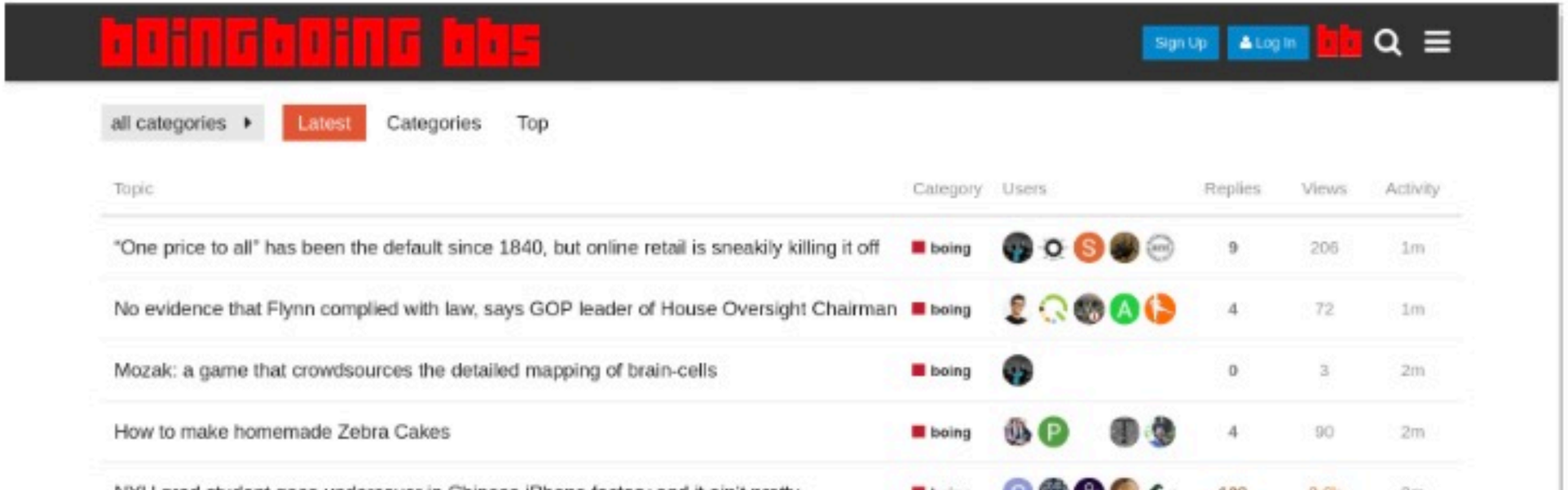
Discourse

Discourse is the 100% open source discussion platform built for the next decade of the Internet. Use it as a:

- mailing list
- discussion forum
- long-form chat room

To learn more about the philosophy and goals of the project, visit **discourse.org**.

Screenshots



Linux

Linux

- At the same time when Swift was open sourced, it was also released for Linux.
- Initial raw support, but has grown a lot since then.
- Swift Package Manager has been fully developed, and has gone through a lot of proposals.
- Official docker image: https://hub.docker.com/r/_/swift/
- XCTest is shipped with Swift: just run `swift test`
- The Server APIs Project

Swift Package Manager

- Bundled with Swift.
- Uses Swift as language to describe dependencies.
- Can generate Xcode project in macOS.
- Only builds on host platforms (thus, no iOS)
- Not currently compatible with Xcode projects for apps.

Swift Package Manager

```
mkdir MyPackage
cd MyPackage
swift package init # or swift package init --type library
# will create 'Sources' and 'Tests' folders
# will create a 'Package.swift' file
swift build
swift test
```

Swift Package Manager

```
import PackageDescription

let urlString = "https://github.com/apple/example-package-playingcard.git"

let package = Package(
    name: "MyPackage",
    dependencies: [
        .Package(url: urlString, majorVersion: 3),
    ]
)

/// Dependencies will go in a 'Packages' folder
```

Resources

Websites and Apps

- swift.org
- apple.github.io/swift-evolution/
- Evolution App
- theswiftwebdeveloper.com

Resources

Mailing Lists

- swiftweekly.com
- swiftweekly.github.io

Resources

Podcasts

- Swift Unwrapped
- Swift Coders

Thank You

https://joind.in/talk/ed577

https://github.com/broomburgo/Getting-started-with-Swift