WHAT'S NEW IN SWIFT 5.1

FEDERICO ZANETELLO

★★★★★ fivestars.blog • @zntfdr

SWIFT EVOLUTION

SE-0068

Expanding Swift Self to class members and value types

Improving use of Swift metatypes

```
class APIManager {
    class var endpoint: String { return "api.dev.startup.com" }

    func printDebugEndpoint() {
        // new in Swift 5.1
        print(Self.endpoint)
    }
}

class APIProductionManager: APIManager {
    override class var endpoint: String { return "api.startup.com" }
}

APIProductionManager().printDebugEndpoint() // prints api.startup.com
```

SE-0240

Ordered Collection Diffing

👋 IGListKit

```
// Swift 5 😂
tableView.reloadData()

// Swift 5.1
let difference = newArray.difference(from: oldArray)

//                                  custom convenience method 👇
let (deletedIndexPaths, insertedIndexPaths) = computeIndexPaths(from: difference)

tableView.performBatchUpdates({ [weak tableView] in
  tableView?.deleteRows(at: deletedIndexPaths, with: .automatic)
  tableView?.insertRows(at: insertedIndexPaths, with: .automatic)
})
```

SE-0242

Synthesize default values for the memberwise initializer

Remove all the inits!

```
struct Dog {
    var age: Int = 0
    var name: String

    // Synthesized initializer in Swift 5
    init(age: Int, name: String)

    // Synthesized initializer in Swift 5.1
    init(age: Int = 0, name: String)
}
```

SE-0244

Opaque Result Types

Easy Type Erasure

```
protocol Shape { /*...*/ }

// Functions returning `some ..` will *always* return the same
// type of instances, without leaking details on which type exactly.
// This is guaranteed at *compile time*
func f() -> some Shape {
  return ...
}

// We can now use generic protocols as return types
func f2() -> some Equatable {
    return ...
}
```

SE-0245

Add an Array Initializer with Access to Uninitialized Storage

Create arrays with specified capacity

```
//  (up to) how many elements the array is going to be 👇
let randomNumbers = [Int](unsafeUninitializedCapacity: 10) {
    buffer, initializedCount in
    for i in 0..<10 { buffer[i] = Int.random(in: 0...5) }

    // set 👇 with the final number of elements in the array
    initializedCount = 10

    // It's ok to not use all the capacity you've
    // asked for, but don't go over that number 💥
}
```

SE-0247

Contiguous Strings

Improve UTF8 string operation performance and APIs

```
let myString = "Swift 5.1 is awesome 🚀"

// Swift 5: generates utf8 representation first, might fail
myString.utf8.withContiguousStorageIfAvailable { body -> () in
    // prints UTF8 representation
    body.forEach { print($0) }
}

// Swift 5.1: generates *continuos* utf8 if necessary, doesn't fail
myString.withUTF8 { body in
    // prints UTF8 representation
    body.forEach { print($0) }
}
```

SE-0248

String Gaps and Missing APIs

More String, Unicode, Character Improvements

```
guard let airplane = Unicode.Scalar(9992) else { return }
print(airplane) // prints ✈

// prints UTF-8 representation
airplane.utf8.forEach { print($0) } // prints 226, 156, 136

let airplaneWidth = Unicode.UTF8.width(airplane)
print(airplaneWidth) // prints 3

let isASCII = Unicode.UTF8.isASCII(codeUnit)
print(isASCII) // prints true
```

SE-0251

SIMD additions

Remove all your SIMD extensions

```
let onesVector = SIMD4<Int>.one // generates a vector with 1 in all elements
print(onesVector) // prints SIMD4<Int>(1, 1, 1, 1)

let vector = SIMD4<Double>(1.0, 2.0, 4.0, 8.0)
print(vector.min(), vector.max()) // prints 1.0 8.0

let lowerVector = SIMD4<Double>(repeating: 1.5)
let upperVector = SIMD4<Double>(repeating: 4.5)
let clampedVector = vector.clamped(lowerBound: lowerVector, upperBound: upperVector)
print(clampedVector) // prints SIMD4<Double>(1.5, 2.0, 4.0, 4.5)

let sum = vector.sum()
print(sum) // prints 15 (= 1.0 + 2.0 + 4.0 + 8.0)
```

SE-0252

Key Path Member Lookup

Keypath-only dynamicMembers 👀

```
@dynamicMemberLookup struct TwitterUser {
    var displayName: String
    var handle: String

    // new in Swift 5.1
    subscript<U>(dynamicMember keyPath: WritableKeyPath<Self, U>) -> U? {
        return self[keyPath: keyPath]
    }
}

let user = TwitterUser(displayName: "Federico Zanetello", handle: "@zntfdr")
let userHandle = user[dynamicMember: \TwitterUser.handle]
print(userHandle2) // prints Optional("@zntfdr")
```

SE-0254

Static and class subscripts

Subscripts, subscripts everywhere

```
enum HTTPStatusCode: Int, Error {
    case badRequest = 400, unauthorized, /* ... */, notFound // ...

    // new in 5.1
    public static subscript(_ statusCode: Int) -> HTTPStatusCode? {
        return HTTPStatusCode(rawValue: statusCode)
    }
}

let errorStatusCode = HTTPStatusCode[404]!
// errorStatusCode is HTTPStatusCode.notFound
```

SE-0255

Implicit returns from single-expression functions

The less code, the better

```
override func tableView(_ tableView: UITableView,
                        numberOfRowsInSection section: Int)
                        -> Int {
    5
}
// no longer need for `return` in one liner functions 🎉
```

SWIFT REPORT

SR-7799

Matching optional enums against non-optional values

No more `case .x?:`

```
enum PRStatus { case open, close }

let optionalEnum: PRstatus? = .open

switch optionalEnum {
  case .open: break
  case .close: break
  case nil: break
}
```

SR-2688

autoclosure does not support closure typealias

@autoclosure ❤️ typealias

```
// This code doesn't compile in Swift 5
// (but does in 5.1)
class Foo {
  typealias FooClosure = () -> String
  func fooFunc(closure: @autoclosure FooClosure) {}
}
```


& MORE!

MODULE STABILITY

Unlocks Swift framework binaries distribution 🚀

LINKS

Resources:

github.com/apple/swift-evolution
github.com/apple/swift
forums.swift.org
whatsnewinswift.com • @twostraws

Slides Style:

jessesquires.com • @jesse_squires

WHAT'S NEW IN SWIFT 5.1

FEDERICO ZANETELLO

★★★★★ fivestars.blog • @zntfdr