Swift

Daniel Sandoval
daniel@loopec.com.br

Linguagens de Programação

Linguagens de Programação

- Legibilidade
- Simplicidade
- Ortogonalidade e Consistência
- Expressividade
- Abstração

Linguagens de Programação

- Assembly
- C
- C++
- Objective-C
- Swift

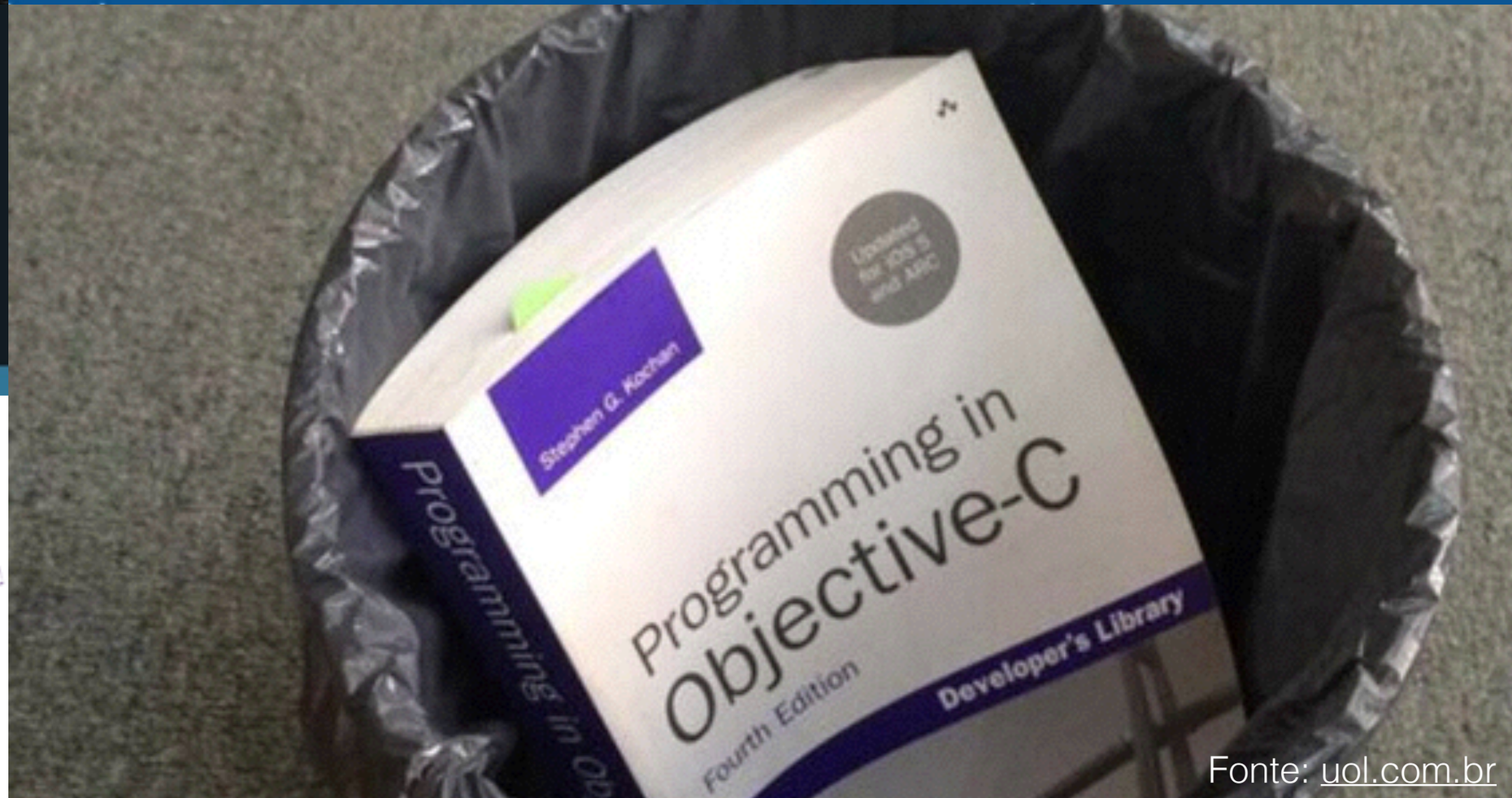
“It is the first industrial-quality systems programming language that is as expressive and enjoyable as a scripting language. […] It’s designed to scale from 'hello, world' to an entire operating system.”

–The Swift Programming Language [3]

Swift

- Iniciado por Chris Lattner [4] em 2010
- Linguagem de alto nível moderna e segura
- Fácil, poderosa e muito eficiente
- *dot syntax*, blocos, ARC, literais e uniformização de interfaces

Fonte: twitter.com



Fonte: uol.com.br

“Swift is the most important change to Apple’s development platform since the switch from PowerPC to Intel processors. […] Swift is a significant step up in terms of safety, expressiveness, features, and perhaps even performance, all in its 1.0 release.”

–John Siracusa [2]

```
println("Hello, world!")
```

,


BRASILIA
.br
COCOAHEADS

```swift
let constanteCidade = "Brasília"
var variavelEvento = "4º CocoaTalks"

   variavelEvento += " " + constanteCidade
!! constanteCidade = "Belo Horizonte"
```

```swift
let constanteCidade = "Brasília"
var variavelEvento = "4º CocoaTalks"

variavelEvento += " " + constanteCidade
```

```
variavelEvento
String variavelEvento
```

Tipos

- Linguagem fortemente tipada
- Inferência de tipos

```swift
let implicitInteger = 26
```

```
let implicitInteger = 26
let implicitDouble = 26.0
```

```
let implicitInteger = 26
let implicitDouble = 26.0
let explicitDouble : Double = 26
```

```
let implicitInteger = 26

let dateLabel = "Hoje é dia " + implicitInteger + "!"
```

```swift
let implicitInteger = 26

let dateLabel = "Hoje é dia \(implicitInteger)!"
```

```
let gruposCocoaHeads = ["Brasília",
    "Porto Alegre",
    "Fortaleza",
    "Campinas",
    "São Paulo"]
```

```
let eventosHoje = ["Brasília" : (19, "Auditório-CDT"),
    "São Paulo" : (19, "Microsoft Brasil"),
    "Campinas" : (19, "Ci&T Campinas"),
    "Porto Alegre" : (19, "Faculdade de Tecnologia Senac - RS")]
```

```swift
let gruposCocoaHeads = ["Brasília",
    "Porto Alegre",
    "Fortaleza",
    "Campinas",
    "São Paulo"]

let eventosHoje = ["Brasília" : (19, "Auditório-CDT"),
    "São Paulo" : (19, "Microsoft Brasil"),
    "Campinas" : (19, "Ci&T Campinas"),
    "Porto Alegre" : (19, "Faculdade de Tecnologia Senac - RS")]
```

```
eventosHoje
V  [String : (Int, String)] eventosHoje
```

```swift
let gruposCocoaHeads = ["Brasília",
    "Porto Alegre",
    "Fortaleza",
    "Campinas",
    "São Paulo"]

let eventosHoje = ["Brasília" : (19, "Auditório-CDT"),
    "São Paulo" : (19, "Microsoft Brasil"),
    "Campinas" : (19, "Ci&T Campinas"),
    "Porto Alegre" : (19, "Faculdade de Tecnologia Senac - RS")]

for i in 0..<gruposCocoaHeads.count {
    var grupo = gruposCocoaHeads[i]
    if let (horario, local) = eventosHoje[grupo] {
        println("O CocoaHeads \(grupo) tem evento hoje às \(horario)h em \(local)")
    } else {
        println("O CocoaHeads \(grupo) não tem evento hoje. 😒")
    }
}
```

```swift
let gruposCocoaHeads = ["Brasília",
    "Porto Alegre",
    "Fortaleza",
    "Campinas",
    "São Paulo"]

let eventosHoje = ["Brasília" : (19, "Auditório-CDT"),
    "São Paulo" : (19, "Microsoft Brasil"),
    "Campinas" : (19, "Ci&T Campinas"),
    "Porto Alegre" : (19, "Faculdade de Tecnologia Senac - RS")]


for grupo in gruposCocoaHeads {
    if let (horario, local) = eventosHoje[grupo] {
        println("O CocoaHeads \(grupo) tem evento hoje às \(horario)h em \(local)")
    } else {
        println("O CocoaHeads \(grupo) não tem evento hoje. 😒")
    }
}
```

```swift
let gruposCocoaHeads = ["Brasília",
    "Porto Alegre",
    "Fortaleza",
    "Campinas",
    "São Paulo"]

let eventosHoje = ["Brasília" : (19, "Auditório-CDT"),
    "São Paulo" : (19, "Microsoft Brasil"),
    "Campinas" : (19, "Ci&T Campinas"),
    "Porto Alegre" : (19, "Faculdade de Tecnologia Senac - RS")]


for grupo in gruposCocoaHeads {
    if let (horario, local) = eventosHoje[grupo] {
        println("O CocoaHeads \(grupo) tem evento hoje às \(horario)h em \(local)")
    } else {
        println("O CocoaHeads \(grupo) não tem evento hoje. 😒")
    }
    switch grupo {
        case "Brasília":
            println("O CocoaHeads Brasília é o melhor!")
        default:
            println("meh..")
    }
}
```

Optionals

- Um valor pode existir ou não
- Ponteiros para `nil`?
- Certeza de que um valor existe quando não é um Optional

```
var usuarioObrigatorio = "Daniel"
```

```swift
var usuarioObrigatorio = "Daniel"
var usuarioOpcional : String?
if let nome = usuarioOpcional {
    println("Boa noite, \(nome)!")
} else {
    println("Boa noite!")
}
```

```swift
var usuarioObrigatorio = "Daniel"
var usuarioOpcional : String? = usuarioObrigatorio
if let nome = usuarioOpcional {
    println("Boa noite, \(nome)!")
} else {
    println("Boa noite!")
}
```

```swift
var usuarioObrigatorio = "Daniel"
var usuarioOpcional : String? = usuarioObrigatorio

if let nome = usuarioOpcional {
    println("Boa noite, \(nome)!")
} else {
    println("Boa noite!")
}

usuarioOpcional = nil
!!usuarioObrigatorio = nil
```

```swift
var usuarioObrigatorio = "Daniel"
var usuarioOpcional : String? = usuarioObrigatorio

if let nome = usuarioOpcional {
    println("Boa noite, \(nome)!")
} else {
    println("Boa noite!")
}

usuarioOpcional = nil
!!usuarioObrigatorio = nil

var valorObrigatorio : String
!!println("\(valorObrigatorio)")
```

```swift
var usuarioObrigatorio = "Daniel"
var usuarioOpcional : String? = usuarioObrigatorio
println("\(usuarioOpcional)")
// Optional("Daniel")
```

```swift
var usuarioObrigatorio = "Daniel"
var usuarioOpcional : String? = usuarioObrigatorio
println("\(usuarioOpcional)")
// Optional("Daniel")
println("\(usuarioOpcional!)")
// Daniel
var USUARIOOPCIONAL = usuarioOpcional?.uppercaseString    //String?
var USUARIOOBRIGATORIO = usuarioOpcional!.uppercaseString //String
```

Funções

- Uma função é um tipo de primeira classe
- Parâmetros nomeados ou não
- Funções aninhadas

```swift
func calculaMedia(a :Double, b :Double) -> Double {
    return (a+b)/2.0
}

calculaMedia(20, 30) // 25.0
```

```swift
func calculaMedia(a :Double, b :Double) -> Double {
    return (a+b)/2.0
}

calculaMedia(20, 30) // 25.0

func calculaMedia(valores :Double...) -> Double {
    var media = 0.0
    for valor in valores {
        media += valor
    }
    return media/Double(valores.count)
}

calculaMedia(20, 30, 40) // 30.0
```

```swift
func soma(#este: Int, #comEste: Int) -> Int {
    return este+comEste
}

soma(este: 20, comEste: 30) // 50
```

```swift
func somaEste(primeiro: Double, comEste segundo: Double) -> Double {
    return primeiro+segundo
}

somaEste(20.0, comEste: 30.0) // 50
```

```swift
func incrementa(valor: Int, incremento: Int) -> Int {
    var valorParaAdicionar = 0
    func fazModuloIncremento() {
        valorParaAdicionar = abs(incremento)
    }
    fazModuloIncremento()
    return valor+valorParaAdicionar
}

incrementa(10, 20)
```

```swift
func ehPalindromo(valor: String) -> Bool {
    return (Array(valor) == reverse(valor))
}

func filtraComFuncao(valores: [String], filtro: (String -> Bool)) -> [String] {
    var resultado = [String]()
    for valor in valores {
        if (filtro(valor)) {
            resultado.append(valor)
        }
    }
    return resultado
}

var palavras = ["salta o atlas",
    "arara",
    "acasacedeopardaloiratoesiofaleairavelaradosserrotes" +
    "etorressodaralevariaelafoiseotarioladrapoedecasaca",
    "adrogadodoteetododagorda",
    "arroz e zorra",
    "Este não é um palíndromo"]

var palindromos = filtraComFuncao(palavras, ehPalindromo)
```

```swift
func ehPalindromo(valor: String) -> Bool {
    return (Array(valor) == reverse(valor))
}

func temNumeroDeLetrasPar(valor: String) -> Bool {
    return (countElements(valor)%2 == 0)
}
```

```swift
func escolheFiltro(num : Int) -> ((String) -> Bool) {
    switch num {
    case 1:
        return temNumeroDeLetrasPar
    default:
        return ehPalindromo
    }
}
```

```swift
var numeros = [Int](1...10)
var multiplicado = numeros.map({ (numero: Int) -> Int in
    return 10*numero
})
```

```swift
var numeros = [Int](1...10)
var multiplicado = numeros.map({ (numero: Int) -> Int in
    return 10*numero
})

multiplicado = numeros.map({ numero in 10*numero })
```

```swift
var numeros = [Int](1...10)
var multiplicado = numeros.map({ (numero: Int) -> Int in
    return 10*numero
})

multiplicado = numeros.map({ numero in 10*numero })

multiplicado = numeros.map({ 10*$0 })
```

```swift
var numeros = [Int](1...10)
var multiplicado = numeros.map({ (numero: Int) -> Int in
    return 10*numero
})
```

```swift
multiplicado = numeros.map({ numero in 10*numero })

multiplicado = numeros.map({ 10*$0 })

multiplicado = numeros.map { 10*$0 }
```

```swift
class FormaGeometrica {

    var numeroDeLados = 0

    var nome : String

    var description : String {
        get {
            return "A figura geométrica \(nome) tem \(numeroDeLados) lados."
        }
    }

    init(nome : String) {
        self.nome = nome
    }

    convenience init(nome : String, numeroDeLados: Int) {
        self.init(nome: nome)
        self.numeroDeLados = numeroDeLados
    }

}

var quadrado = FormaGeometrica(nome: "Meu primeiro quadrado", numeroDeLados: 4)
println(“\(quadrado.description)")
// A figura geométrica Meu primeiro quadrado tem 4 lados.
```

```swift
class FormaGeometrica {

    var numeroDeLados = 0

    var nome : String

    var description : String {
        get {
            return "A figura geométrica \(nome) tem \(numeroDeLados) lados."
        }
    }

    init(nome : String) {
        self.nome = nome
    }

    convenience init(nome : String, numeroDeLados: Int) {
        self.init(nome: nome)
        self.numeroDeLados = numeroDeLados
    }

}

var quadrado = FormaGeometrica(nome: "Meu primeiro quadrado", numeroDeLados: 4)
println(“\(quadrado.description)")
// A figura geométrica Meu primeiro quadrado tem 4 lados.
```

```
enum Rank : Int {
    case Ace = 1
    case Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten
    case Jack, Queen, King

    func description() -> String {
        switch self {
        case .Ace:
            return "Ace"
        case .Jack:
            return "Jack"
        case .Queen:
            return "Queen"
        case .King:
            return "King"
        default:
            return String(self.rawValue)
        }
    }
}
```

```swift
enum Naipe : String {
    case Copas = "❤️"
    case Espadas = "♠️"
    case Ouro = "♦️"
    case Paus = "♣️"

    func description() -> String {
        return self.rawValue
    }
}
```

```swift
struct Carta {
    var rank : Rank
    var naipe : Naipe

    func description() -> String {
        return rank.description() + naipe.description()
    }
}

var carta = Carta(rank: .Ace, naipe: .Espadas)
println(carta.description())
//Ace♠️
```

Classes e Structs

- Uma classe é sempre passada por referência
- Structs são tipos mais complexos
- Structs são passadas por referência, mas copiadas quando modificadas

```swift
protocol Descritivel {
    func description() -> String
}

struct Carta : Descritivel {
    var rank : Rank
    var naipe : Naipe

    func description() -> String {
        return rank.description() + naipe.description()
    }
}

enum Rank : Int, Descritivel {
...
}

enum Naipe : String, Descritivel {
...
}
```

```swift
extension String {
    func cpfFormatted() -> String {
        return self.substringToIndex(advance(self.startIndex, 3)) + "." +
        self.substringWithRange(Range<String.Index>(start: advance(self.startIndex, 3),
            end: advance(self.startIndex, 6))) + "." +
        self.substringWithRange(Range<String.Index>(start: advance(self.startIndex, 6),
            end: advance(self.startIndex, 9))) + "-" +
        self.substringWithRange(Range<String.Index>(start: advance(self.startIndex, 9),
            end: advance(self.startIndex, 11)))
    }
}

"12345678901".cpfFormatted()
```

“In an audit of the last 3 years of shipped bugs in our robotics software, ~40% would have been caught early by using Swift.”

–Brad Larson, SonoPlot Inc. [1]

Obrigado!

Daniel Sandoval
daniel@loopec.com.br

Referências

1. http://www.sunsetlakesoftware.com/2014/12/02/why-were-rewriting-our-robotics-software-swift

2. http://arstechnica.com/apple/2014/10/os-x-10-10/21/

3. https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/

4. http://nondot.org/sabre/

5. http://www.slideshare.net/giordano/a-swift-introduction-to-swift