Swift

By Ben Wordekemper, Ryah Oh, and Leo Hoffman

What is Swift?

- Swift is made for mobile app development on Apple and Android
- It was developed by Apple to replace Objective-C
- First Introduced it in 2014 at the Worldwide Developers Conference.
- Open source

Apps Made with Swift

- Lyft
- LinkedIn
- AirBnB
- Khan Academy
- Slack

Goals of Swift

- Apple used objective-C for years, a language that was notoriously complex and difficult to read.
- A simpler language was needed to create the same apps and was easier to read.
- Apple wanted the language to share the object-oriented features of Objective-C, and would be hard to crash.
- They used techniques that would be better at catching bugs and would address common programming errors.

Key Features - Design

Modern and Powerful

- Combines elements from several programming languages
- Syntax and features aimed at ease of use and safety

Concise Syntax

- Minimizes boilerplate code
- Implements functional programming paradigms

Key Features - Safety and Performance

- Strongly typed language
    - Avoids type incompatibilities
- Automatic memory management
    - Automatic garbage collection using reference counting
- High-performance compiler

Safety Features:

- Safely handle nil values
- Avoid null pointer errors

Deep Dive into Swift

Closures

- Powerful tool for managing functionality
- Encourages functional programming patterns

Generics

- Enables writing flexible, reusable functions and types
- Allows type-safe collections and algorithms

Pass by Value vs Pass by Reference

- Classes are passed by references
- By default, other objects are passed by value
    - Default can be overridden using the inout keyword.

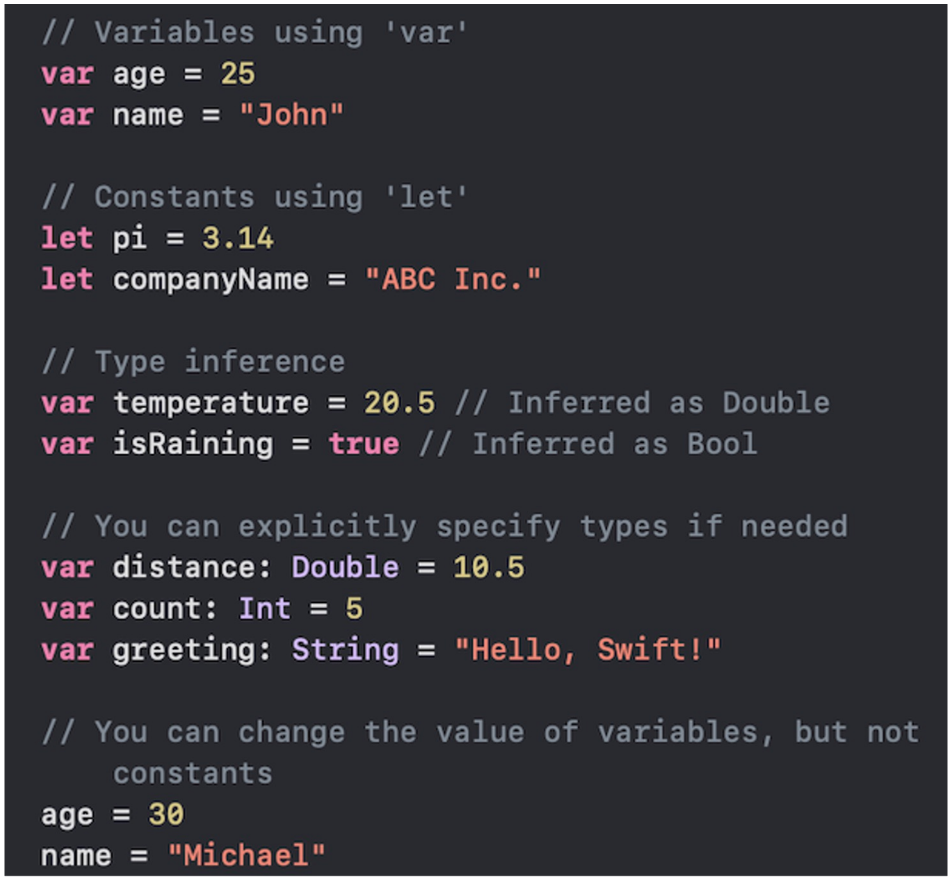
Overview and Syntax

- Uses LLVM compiler technology, which draws inspiration from Objective-C, Rust, Haskell, Pythi, C#, and more.
- Converts to machine code for future proofing
- Apple advertises it as a “Great first language”
- Swift has a concise syntax making it easy to read and write

Syntax Overview

Variables and Constants

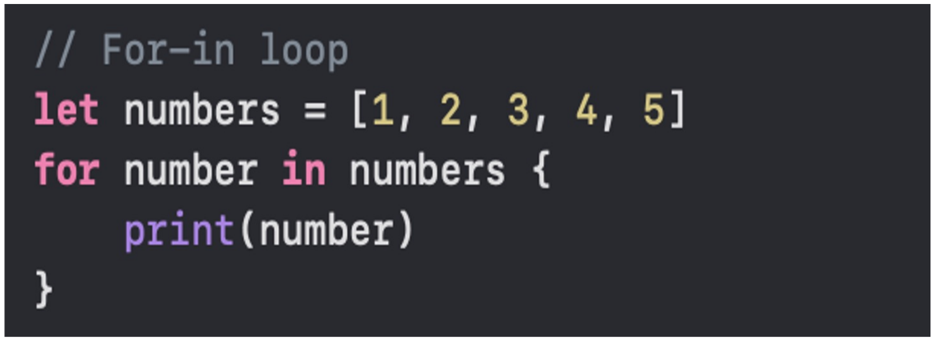
- var for variables
- let for constants



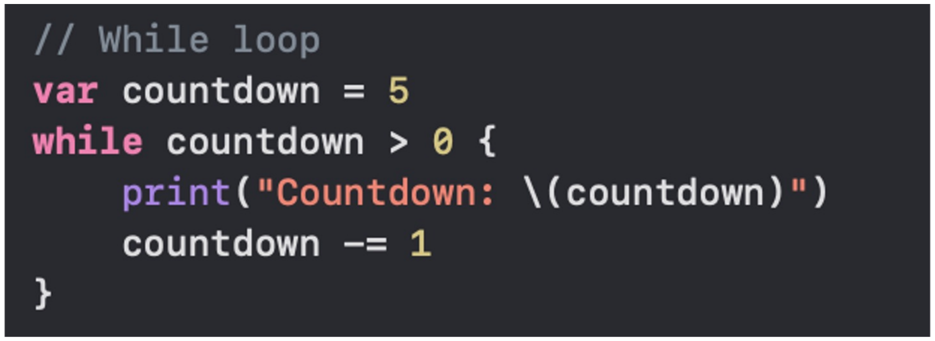
```swift
// Variables using 'var'
var age = 25
var name = "John"

// Constants using 'let'
let pi = 3.14
let companyName = "ABC Inc."

// Type inference
var temperature = 20.5 // Inferred as Double
var isRaining = true // Inferred as Bool

// You can explicitly specify types if needed
var distance: Double = 10.5
var count: Int = 5
var greeting: String = "Hello, Swift!"

// You can change the value of variables, but not
    constants
age = 30
name = "Michael"
```

Syntax Overview

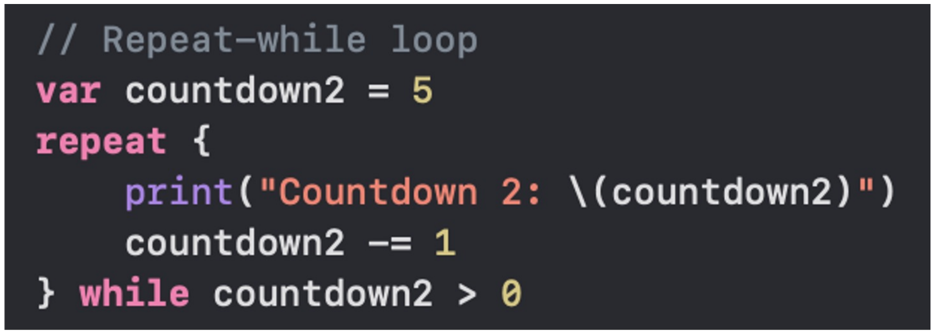
Control Flow



```swift
// For-in loop
let numbers = [1, 2, 3, 4, 5]
for number in numbers {
    print(number)
}
```

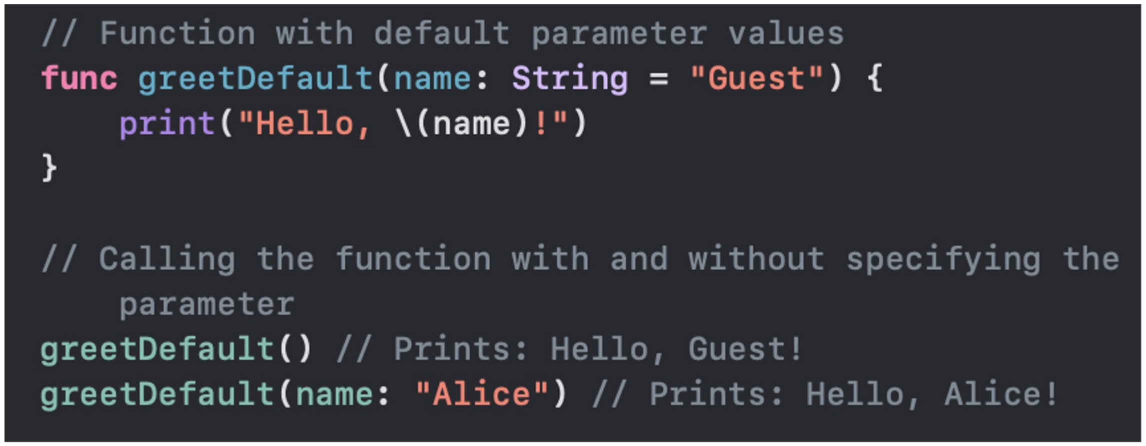


```swift
// While loop
var countdown = 5
while countdown > 0 {
    print("Countdown: \(countdown)")
    countdown -= 1
}
```

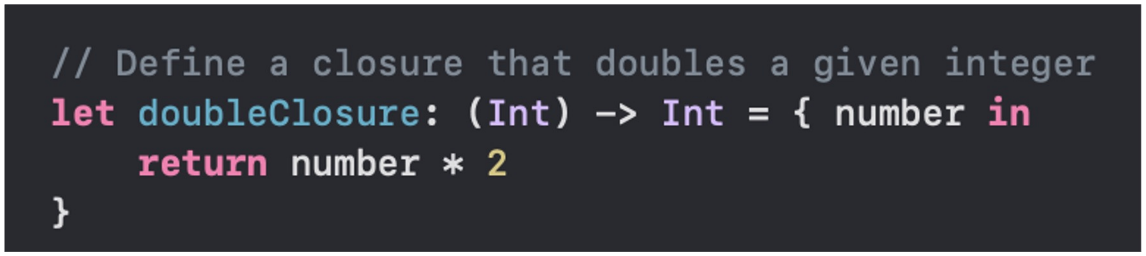


```swift
// Repeat-while loop
var countdown2 = 5
repeat {
    print("Countdown 2: \(countdown2)")
    countdown2 -= 1
} while countdown2 > 0
```

Syntax Overview



```
// Function with default parameter values
func greetDefault(name: String = "Guest") {
    print("Hello, \(name)!")
}

// Calling the function with and without specifying the
    parameter
greetDefault() // Prints: Hello, Guest!
greetDefault(name: "Alice") // Prints: Hello, Alice!
```



```
// Define a closure that doubles a given integer
let doubleClosure: (Int) -> Int = { number in
    return number * 2
}
```

- A closure gives access to an outer function’s scope from an inner

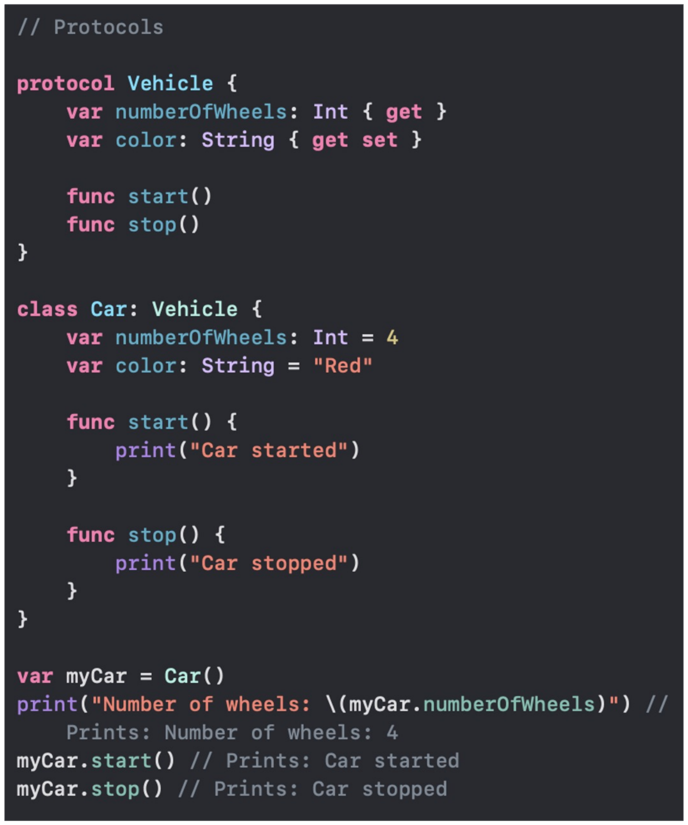
Syntax Overview

Protocol

- Set of rules for behavior that types can conform to
- Promoting code abstraction and reusability.

Class

- Creating objects with reference semantics
- supporting inheritance
- For building object-oriented systems.
- Simpler than interface in Java

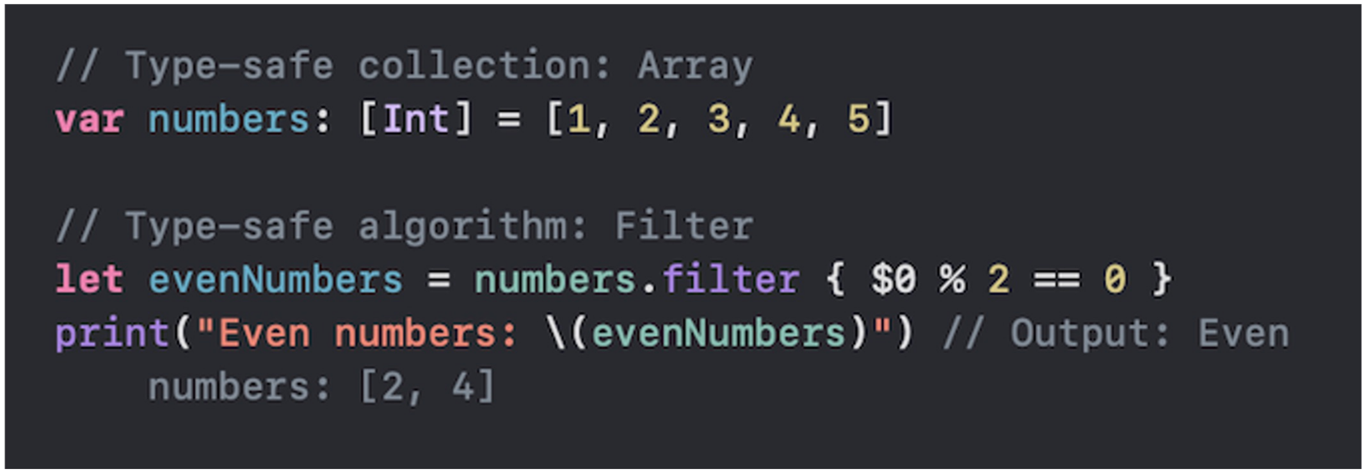


```swift
// Protocols

protocol Vehicle {
    var numberOfWheels: Int { get }
    var color: String { get set }

    func start()
    func stop()
}

class Car: Vehicle {
    var numberOfWheels: Int = 4
    var color: String = "Red"

    func start() {
        print("Car started")
    }

    func stop() {
        print("Car stopped")
    }
}

var myCar = Car()
print("Number of wheels: \(myCar.numberOfWheels)") //
    Prints: Number of wheels: 4
myCar.start() // Prints: Car started
myCar.stop() // Prints: Car stopped
```

Syntax Overview

Type-safe

- Ability to Type-safe allows developers to write code that is both safe and expressive
- Leverage the type system to catch errors at compile-time and prevent runtime issues.

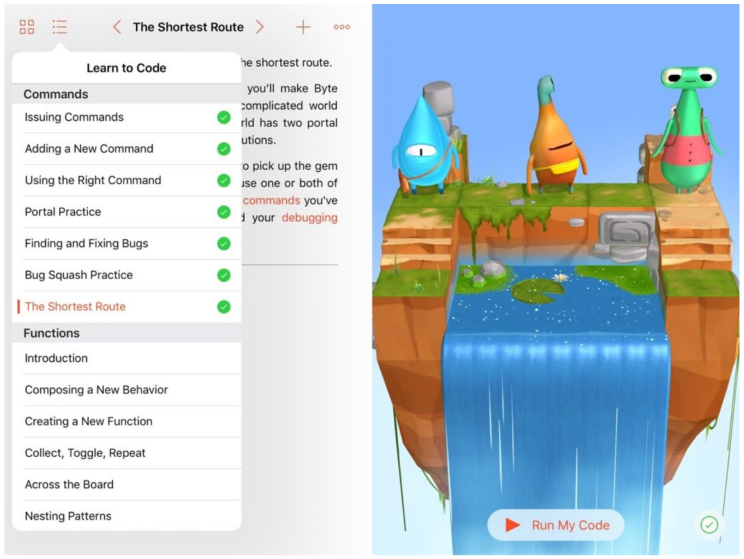


```swift
// Type-safe collection: Array
var numbers: [Int] = [1, 2, 3, 4, 5]

// Type-safe algorithm: Filter
let evenNumbers = numbers.filter { $0 % 2 == 0 }
print("Even numbers: \(evenNumbers)") // Output: Even
    numbers: [2, 4]
```

Overview

Swift offers an exciting journey of learning an innovation

Swift has fulfilled Apple’s goals of creating a language that is easy to use and hard to produce errors.



Demo - Tic Tac Toe

Thank You

Any Questions?

Sources

- https://developer.apple.com/swift/
- https://www.swift.org/
- https://ashfurrow.com/blog/why-objective-c-is-hard-to-learn/
- https://twitter.com/amos_gyamfi/status/1646469566116114432
- https://infostride.com/apps-built-with-swift/
- https://t3.ftcdn.net/jpg/04/37/88/86/360_F_437888641_XrjjuAwATXWNx10jQurCDaXXJnobhDi4.jpg
- https://icon-icons.com/icon/airbnb-tile-logo/168680
- https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQDyddU95DtsSzvMRQ5sjZBQYa3Q8bhYc_RiJu6LcDL&s
- https://stackoverflow.com/questions/27364117/is-swift-pass-by-value-or-pass-by-reference