SCIENCES SORBONNE UNIVERSITÉ

Swift, minimal survival guide

Fabrice.Kordon@lip6.fr

LIP6

As an introduction…

History

- Elaborated since 2010 (secretly)
- ß launched on June 14, 2014 (WWDC)
- Version history
  - **September 2014** — **1.0 (totally unstable)**
  - **September 2015** — **2.0 (towards stability)**
  - **September 2016** — **3.0 (mature)**
  - **September 2017** — **4.0 (backward compatibility concerns)**
  - **September 2018** — **4.2 (nobody mentions it in the Keynote)**

We will see

- Principles
- Main syntactical constructions
- Strings
  - **You will need it rapidly**

Principles

- **Mix of objects and functional traits**
  - Strong trend
- **Includes the most recent compilation facilities**
  - «Easy» and «Compact»
    - **Numerus implicit things**
    - **Many deductions from expressions**
    - **Is it always easy to read?**
- **Inspired from Objective-C**
  - Integration of «external mechanisms»
    - **Automatic Reference Counting**
- **Same foundations than Objective-C**
  - Rely on the same APIs & libraries
  - Easy to pass from one to another

Classes in Swift

By an example

```
class Counter {
    var count = 0
    static var scount = 0

    func add(n: Int) {
        count += n
    }

    func val () -> Int {
        return count
    }

    static func reset() { // satic method
        scount = 0
    }

    class func oulala() -> String { // class method
        return "Oulala"
    }
}
```

More later...

Method invocation

Traditional «dotted» approach

```
class Counter {
    var count = 0
    static var scount = 0

    func add(n: Int) {
        count += n
    }

    func val () -> Int {
        return count
    }

    static func reset() { // satic method
        scount = 0
    }

    class func oulala() -> String { // class method
        return "Oulala"
    }
}

var myValue = Counter()
myValue.add(n: 2)
Counter.reset()
print("\(myValue.val()) \(Counter.oulala())")
```

Produce a trace on the console

print

- Parameters : a string + terminator
- Concatenations/substitutions natively handled (seen later)

Example

```
print("toto is ", terminator:"happy ")
print(" and titi too")
print("I saw a pussy cat")
```

```
toto is happy  and titi too
I saw a pussy cat
```

Strings

No explicit declaration required (type is deduced)

- Thank to dynamic typing
- Support of Unicode
- Two types of variables in Swift
  - **Immutable or mutable**

Examples

```
let immutable   = "Toto the immobile one"
var mutable     = "I may change"
let someChinese = "Hànzi 漢字"
```

String manipulation

- Constants (immutable) declared with «let»
- Variables (mutable) declared with «var»

```
let meteo = "nice"
let msg = "Weather is "+meteo

var logParis = MeteoClass()
print("Wind speed is \(logParis.wind) km/m: ",
    "\(logParis.equivBeaufort(logParis.wind))",
    " on the beaufort scale")

let myString = "Hello"
var fullString : String
fullString = myString+" World!"
```

- To get more...

As a conclusion…

- You (almost 😉) have basics to build your first Swift application alone