UE LI385

Parcours DANT

Introduction to iOS development with Swift

Swift

Adrien Humilière
Trainline

adrien.humiliere@trainline.com

Swift

Swift 1.0 (sep. 2014)

Swift 1.2 (apr. 2015)

Swift 2.0 (sep. 2015)

Swift 3.0 (sep. 2016)

Swift 4.0 (sep. 2017)

Swift 3.0

```
print("Hello, World!")
```

Variables

Variables

```
var myVariable = 42

myVariable = 50
```

Variables

```
let myConstant = 42

myConstant = 50
// Cannot assign to 'let' value 'maConstante'
// myConstant = 50
// ~~~~~~~~~~ ^
```

Variables

```
var myInt = 42
var myFloat = 4.1
var myString = "DANT"

myString = 17

// Cannot assign a value of type 'Int' to a value
// of type 'String'
// myString = 17
//            ^
```

Variables

```
var myInt: Int = 42
var myFloat: Float = 4.1
var myString: String = "DANT"
```

Variables

```
var entierImplicite = 42 // 42
var doubleImplicite = 42.0 // 42.0
var doubleExplicite: Double = 42 // 42.0
```

Optionals

Optionals

```
let x: Float?    // Float or nil
let y: Float     // Float
```

Optionals

```
func myFunction(param: String?) {
    print(param)                    // -> Optional(DANT)

    if let param = param {
        print(param)                // -> DANT
    }
}

myFunction(param: "DANT")
```

Optionals

```
enum Optional <T> {
    case none
    case some(T)
}
```

Optionals

```
let x: String? = nil
let x = Optional<String>.none

let x: String? = "hello"
let x = Optional<String>.some("hello")
```

Tuples

Tuples

```
let x: (String, Int, Double) = ("hello", 5, 0.85)
let (word, number, value) = x

print(word)      // hello
print(number)    // 5
print(value)     // 0.85
```

Tuples

```
func getSize() -> (weight: Int, height: Int) {
    return (250, 80)
}
```

Ranges

Ranges

```
let array = ["a","b","c","d"]
let a = array[2...3]                    // ["c","d"]
let b = array[2..<3]                    // ["c"]


for i in 0..<20 {
    // …
}
```

Data structures

Data structures

`class`

`struct`

`enum`

Data structures: similarities

```
class ViewController {
}

struct CalculatorBrain {
}

enum Op {
}
```

Data structures: similarities

```
func myFunction(arg: Double) -> Int {
}

var storedProperty: Float?
var computedProperty: String {
    get {}
    set {}
}

var storedProperty = 42      // Not enums
```

Data structures: differences

Inheritance (class only)

Value type vs. Reference type

enum, struct

class

Data structures: differences

Value (enum, struct)

- Copied when passed as an argument
- Copied when assigned to a variable
- Immutable if assigne to a variable with let
- Every function that can mutate must have the keyword mutating

Reference (class)

- Can be inherited and can inherit
- Shared in the heap and reference counted
- Constant pointer to a class can still be mutated with methods and properties
- Not copied when passed as an argument

Methods

Methods: params

```
func foo(extFirst first: Int, extSecond second: Double) {
    var sum = 0.0
    for _ in 0..<first { sum += second }
}

foo(extFirst: 123, extSecond: 5.5)
```

Methods: params

```
func foo(_ first: Int, extSecond second: Double) {
    var sum = 0.0
    for _ in 0..<first { sum += second }
}

foo(123, extSecond: 5.5)
```

Methods: params

```
func foo(first: Int, second: Double) {
    var sum = 0.0
    for _ in 0..<first { sum += second }
}

foo(first: 123, second: 5.5)
```

Methods: params

```
func foo(_ first: Int, _ second: Double) {
    var sum = 0.0
    for _ in 0..<first { sum += second }
}

foo(123, 5.5)
```

Methods: keywords

`override`

`final`

`static`

Array

Arrays

```
var a = [String]()
var b: [String]

let animals = ["Dog", "Cow", "Cat"]

animals.append("Sheep")      // Won't build
let animal = animals[3]      // Crash at runtime
```

Arrays

```
for animal in animals {
    print(animal)
}
```

Dictionary

Dictionaries

```
var upmc = [String: Int]()

upmc = ["Students": 1, "Teachers":11]
upmc["Cal"] = 12

let number = upmc["Teachers"]

// ["Cal": 12, "Students": 1, "Teachers": 11]
```

Arrays

```
for (key, value) in upmc {
    print("\(key): \(value)")
}
```

Initialization

Initialization

```
let dant = Classroom()
let dant = Classroom(name: "DANT")
let dant = Classroom(name: "DANT", students: 16)
```

Initialization

```
class Classroom {
    init() {}
    init(name param: String) {}
    init(name param: String, students: Int) {}
}
```

The End.

Resources

adhumi.fr/teaching

Login	Password
`m2sar`	`sarM2`