Adrien Humilière // ahumiliere@captaintrain.com // L3 DANT 2015/2016

Introduction au développement iOS avec Swift

Cours 2 ~ Introduction à Swift

Adrien Humilière // ahumiliere@captaintrain.com // L3 DANT 2015/2016

Aujourd'hui++

Découverte de **Swift** :

- **Syntaxe** de Swift
- **POO** avec Swift
- **Classes, objets, méthodes**
- **Types de données et collections**
- **Qualité** du code

Versions de Swift

- 1.0 (septembre 2014)
- 1.1 (octobre 2014)
  - 1.2 (avril 2015)
- 2.0 (septembre 2015)
- 2.1 (décembre 2015)
- 2.2 (not yet released)

Syntaxe de Swift

Premier programme en Swift

→ **Hello world!** en Swift

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

Premier programme en Swift

➔ **Hello world!** en Java

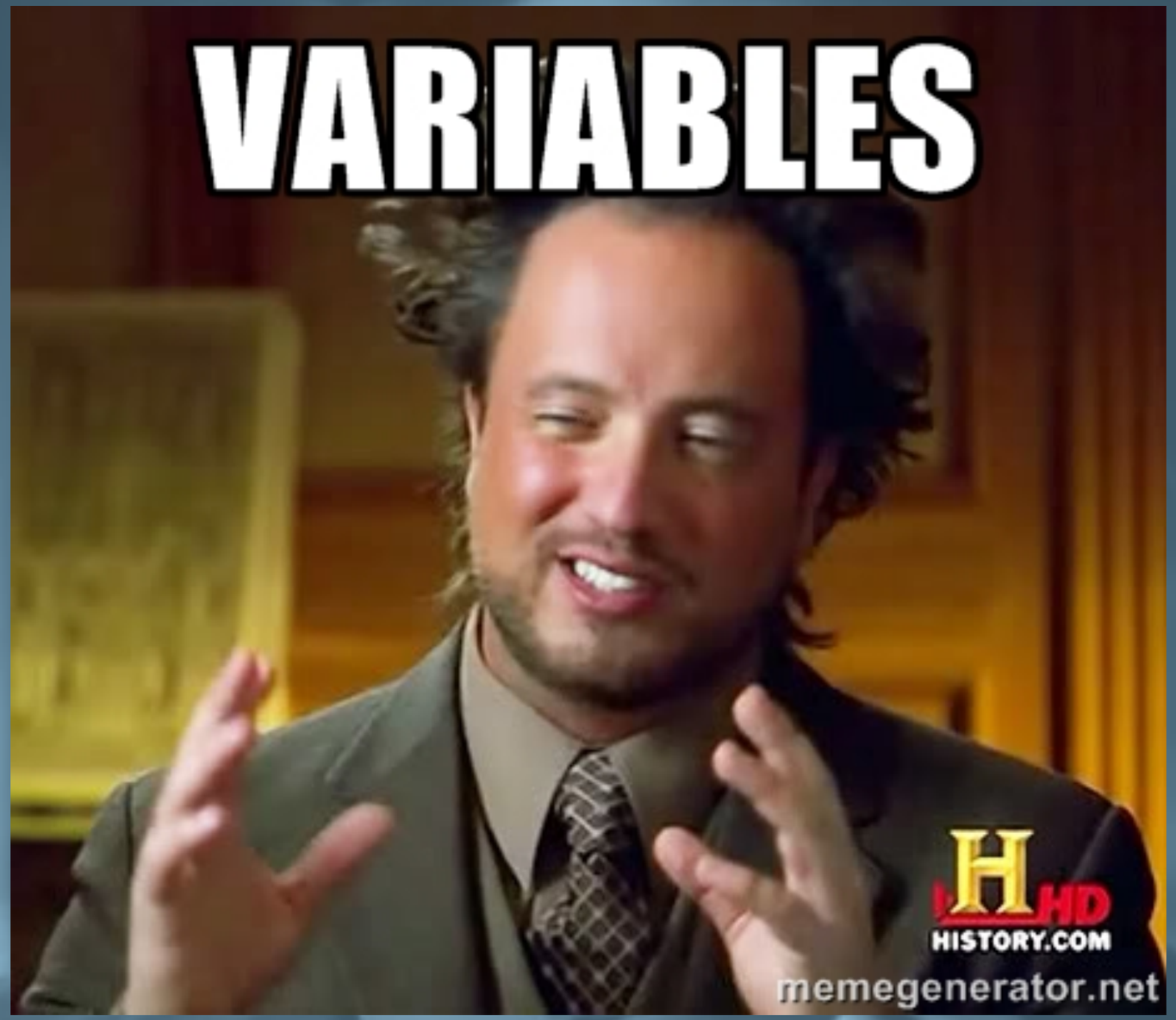
```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Premier programme en Swift

→ Hello world! en Objective-C

```
#import <Foundation/Foundation.h>

int main (int argc, const char * argv[])
{
        NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
        NSLog(@"Hello, World!");
        [pool drain];
        return 0;
}
```



Variables et constantes

→ Déclaration d'une variable

```
var maVariable = 42
```

Variables et constantes

→ Déclaration d'une variable

```
var maVariable = 42
maVariable = 50
```

Variables et constantes

→ Déclaration d'une constante

```
let maConstante = 42
```

Variables et constantes

→ Déclaration d'une constante

```
let maConstante = 42
maConstante = 50
// Cannot assign to 'let' value 'maConstante'
// maConstante = 50
// ~~~~~~~~~~~~ ^
```


LET'S BE HONEST HERE
TYPES ARE USEFUL
memegenerator.net

Types

```
var monInt = 42
var monFloat = 4.1
var monString = "DANT"
```

Types

```
var monInt = 42
var monFloat = 4.1
var monString = "DANT"
monString = 17
```

Types Implicite

```
var monInt = 42
var monFloat = 4.1
var monString = "DANT"
monString = 17
// Cannot assign a value of type 'Int' to a value
// of type 'String'
// monString = 17
//             ^
```

Types Explicite

```
var monInt: Int = 42
var monFloat: Float = 4.1
var monString: String = "DANT"
```

Types Implicite vs. Explicite

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

Types Conversion

```
let label = "La largeur est "
let largeur = 94
let largeurTotale = label + String(largeur)
// La largeur est 94
```

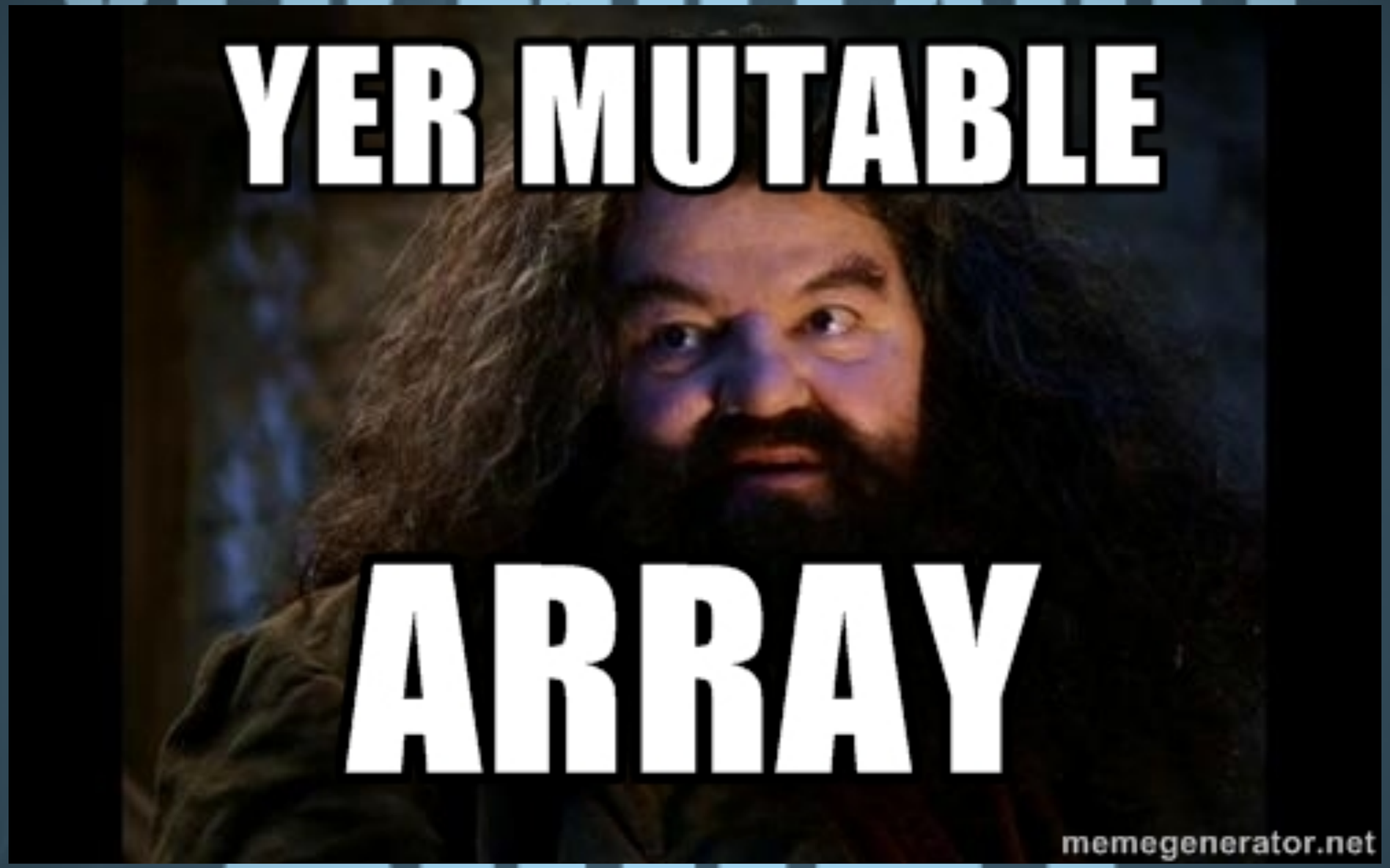
#String

```
let pommes = 3
let oranges = 5
let pommesTotal = "Je possède \(pommes) pommes."
// Je possède 3 pommes.
let orangesTotal = "Je possède \(oranges) oranges."
// Je possède 5 oranges.
let fruitsTotal = "Je possède \(pommes + oranges) fruits en tout."
// Je possède 8 fruits en tout.
let pommesEtOranges = pommesTotal + " " + orangesTotal
// Je possède 3 pommes. Je possède 5 oranges.
```

Types non spécifiques

- `AnyObject` : une instance d'une classe de n'importe quel type
- `Any` : une instance de tout type (y compris une fonction)

```
let array: [AnyObject] = ["DANT", 21, True]
```



Tableaux

```
var tableau = ["Milan", "Leila", "Alban"]
print(tableau[0])
// Milan
```

Tableaux

```
var tableau: [String] = ["Milan", "Leila", "Alban"]
print(tableau[0])
// Milan
```

Tableaux

```
var tableau: [String] = ["Milan", "Leila", "Alban"]
print(tableau[0])
// Milan

tableau[0] = "Khaled"
print(tableau[0])
// Khaled
```

Tableaux

```
let tableau: [String] = ["Milan", "Leila", "Alban"]
tableau[0] = "Xue"
```

Tableaux

```
let tableau: [String] = ["Milan", "Leila", "Alban"]
tableau[0] = "Xue"
// Cannot assign to the result of this expression
// tableau2[0] = "Xue"
// ~~~~~~~~~~~~ ^
```

→ `let` permet de definir un tableau non-mutable

Dictionnaires

→ Même comportement que les tableaux

```
var dictionnaire: [String: String] = [
    "ios": "Adrien Humilière",
    "backend": "Olivier Pitton"
]
dictionnaire["reseaux"] = "Anne Fladenmuller"
```

Tableaux et dictionnaires

Instanciation

```
let tableauVide = [String]()
let dictionnaireVide = [String: Float]()
```


FLOW CONTROL
IT SHALL BE DONE.
USE THE FORCE
memegenerator.net

Control flow

- Conditions : `if` et `switch`
- Boucles : `for-in`, `for`, `while` et `do-while`

Control flow

→ if

```
if moyenne >= 16 {
    println("Très bien")
} else if moyenne >=‰ 10 {
    println("Bien")
} else {
    println("Pas bien")
}
```

Control flow

```
switch legume {
case "celeri":
    println("Ajouter quelques raisins.")
case "concombre", "salade":
    println("Ça devrait faire un bon sandwich.")
case let x where x.hasSuffix("poivre"):
    println("On sent les épices \(x)?")
default:
    println("Tout a bon goût dans la soupe.")
}
```

Control flow

→ C-style for-loop

```
for var i = 0; i < 10; ++i {
    // …
}
```

Control flow

→ C-style for-loop DÉPRÉCIÉ

```
for var i = 0; i < 10; ++i {
    // …
}
```

Control flow

```
for var i = 0; i < 10; ++i {
    // …
}
```

→ Swift **for-in** equivalent

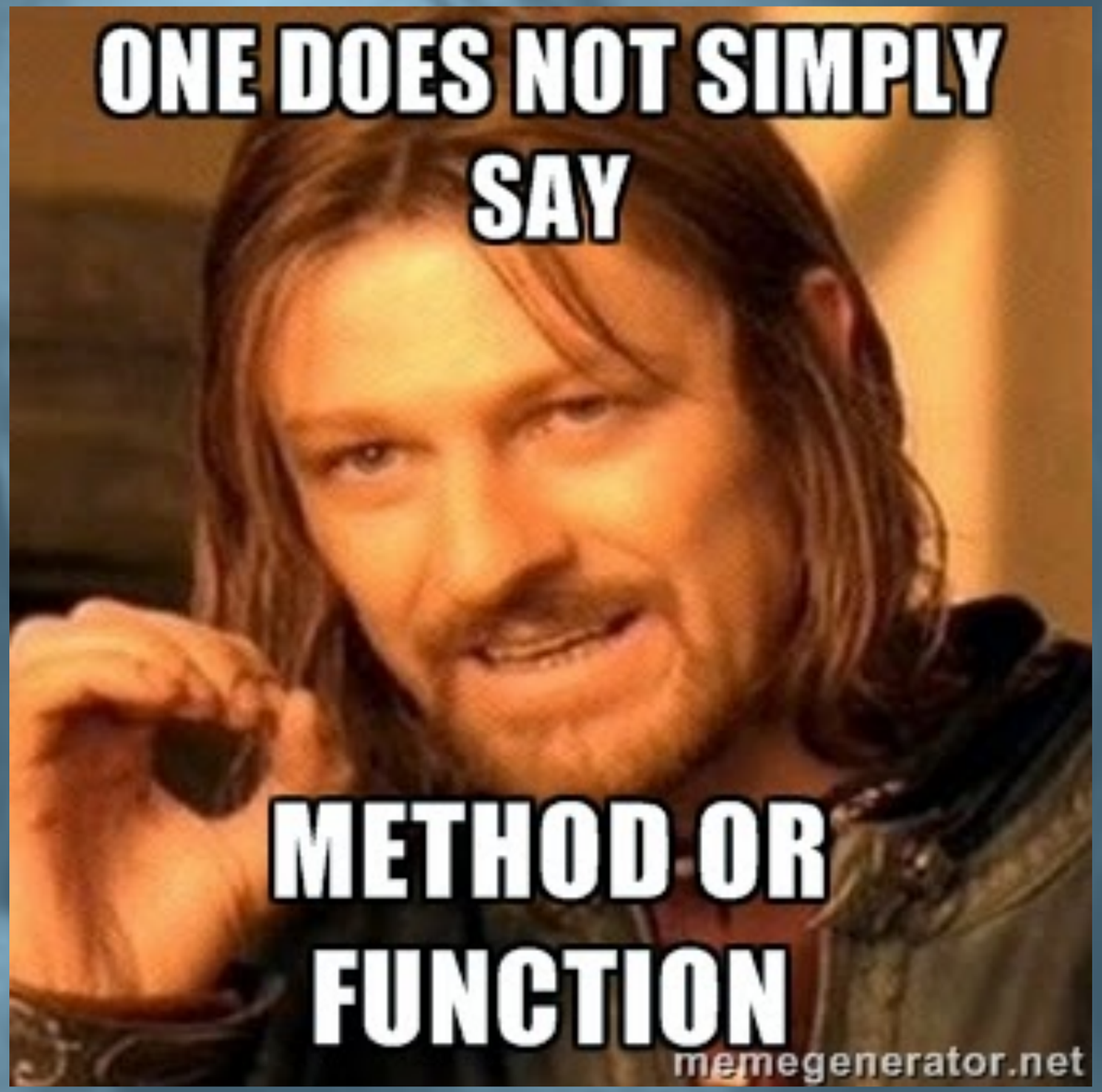
```
for i in 0..<10 {
    // …
}
```

Control flow

→ while & do-while

```
while n < 100 {
    n += n
}

do {
    m += m
} while m < 100
```


ONE DOES NOT SIMPLY
SAY
METHOD OR
FUNCTION
memegenerator.net

Fonctions et méthodes

```
func maFonction(argument: Int) -> String {
    return "L'argument est \(argument)"
}

let resultat = maFonction(2016)
println(resultat)
// L'argument est 2016
```

Objets et classes

```
class MaClasse {
    var maPropriete = "DANT"

    func maMethode(annee: Int) -> [String] {
        // …
    }
}
```

Objets et classes

```
class MaClasse: MaClasseMere {
    var maPropriete = "DANT"

    func maMethode(annee: Int) -> [String] {
        // …
    }
}
```

-> La classe `MaClasse` hérite de `MaClasseMere`

Objets et classes Initialisation

```
class MaClasse {
    var maPropriete: String

    init() {
        maPropriete = "DANT"
    }

    init(name: String) {
        maPropriete = name
    }
}
```

Objets et classes Initialisation

```
class MaClasse {
    var maPropriete: String = "DANT"

    init(name: String) {
        maPropriete = name
    }
}
```

Objets et classes Initialisation

```
let objet = MaClasse()
println(objet.maPropriete)
// DANT

let objet2 = MaClasse(name: "Captain Train")
println(objet2.maPropriete)
// Captain Train
```

The end

TP du jour

Application en ligne de commande.

Voir TP 2 sur adhumi.fr/teaching

Salle Mac

- Login : `m2sar`
- Password : `sarM2`

Ressources (slides, TP, etc.)

- adhumi.fr/teaching