



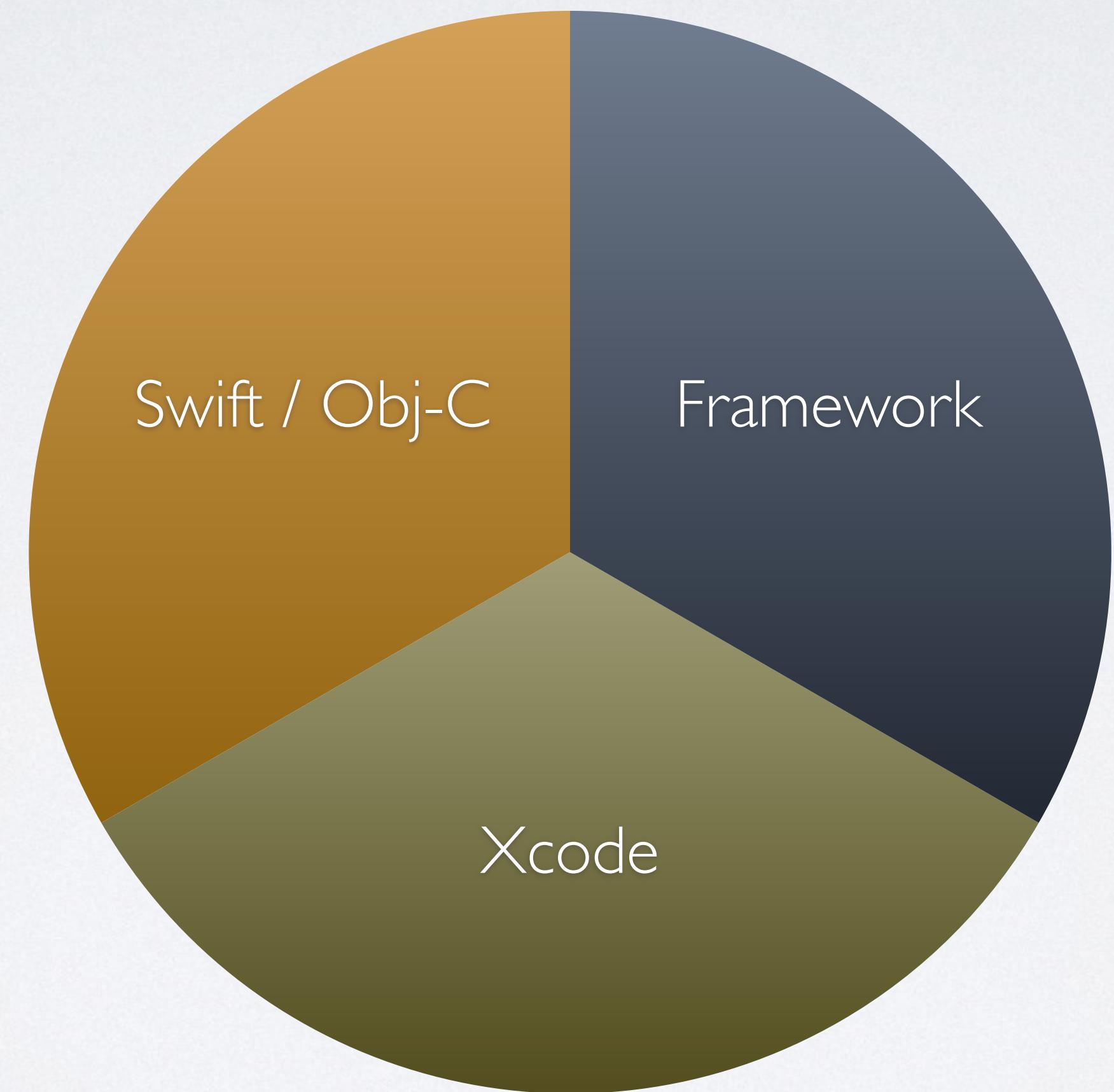
iOS

# PRÉAMBULE

- Initiation à iOS
- Découverte de l'environnement
- Objectifs

Questions ?

# APP IOS



# APP IOS

Swift / Obj-C : Langages de programmation

Framework : iOS, watchOS, macOS, tvOS

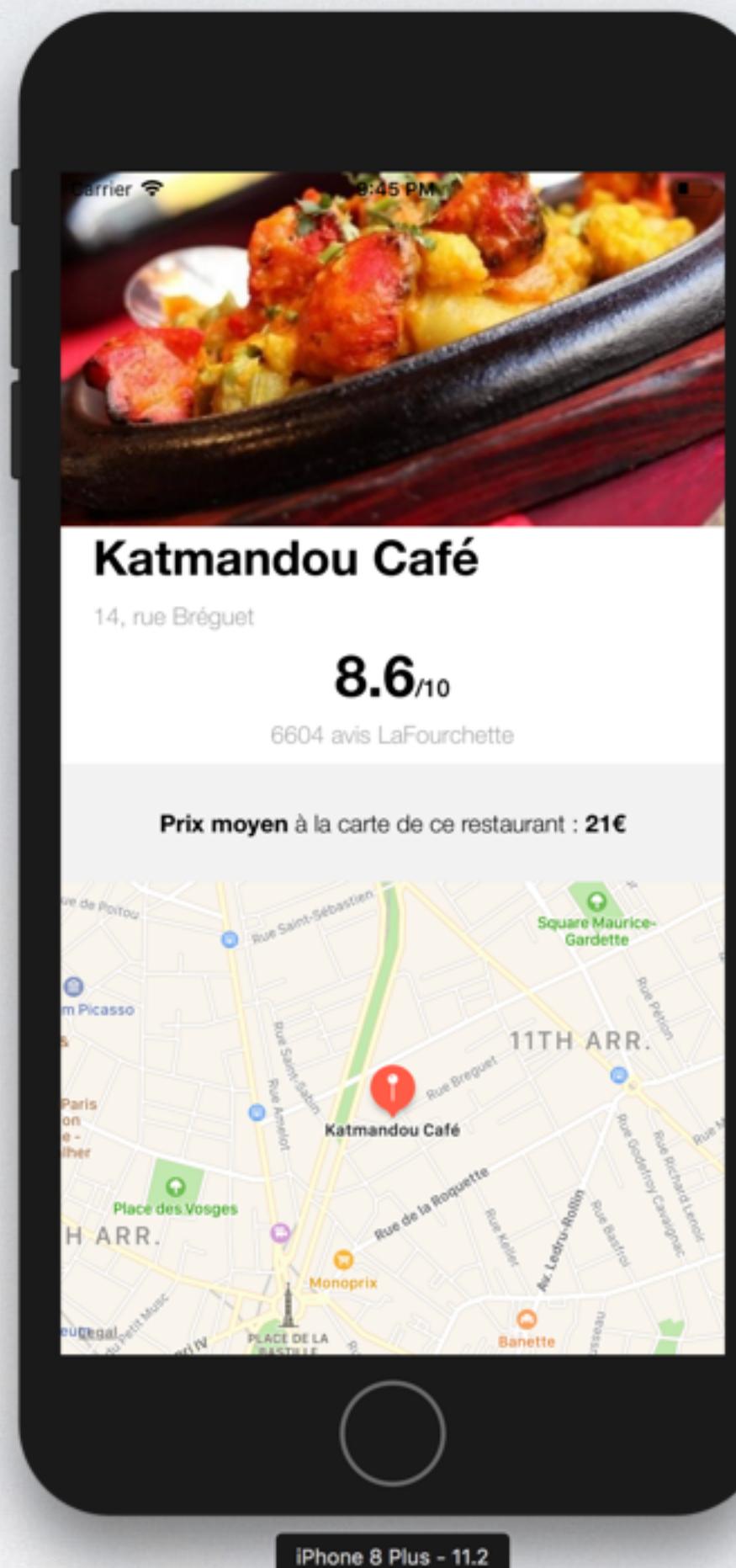
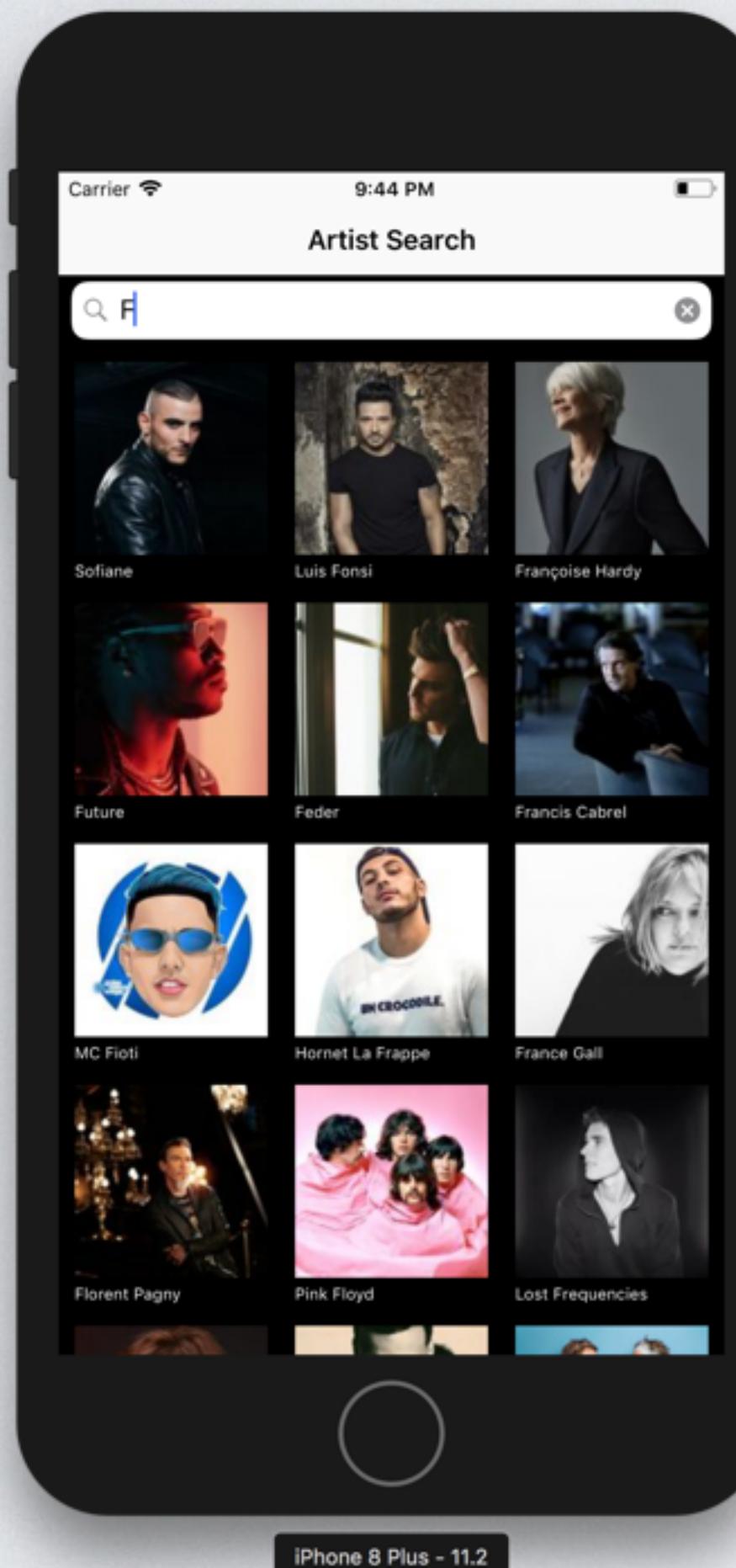
Xcode : Outil de développement IDE



# PRODUCT BACKLOG

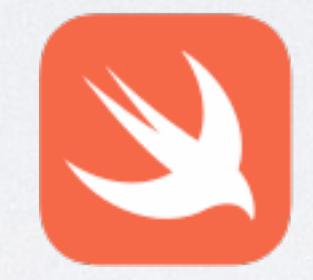
Utiliser Swift : Variables, constantes et opérateurs	Utiliser XCode et Playground Swift	Comprendre MVC	Utiliser des designs patterns
Utiliser Swift : Conditions	Manipuler les fenêtres d'outils XCode	Comprendre les états d'une application	Architecturer votre app
Utiliser Swift : Boucles	Utiliser Interface Builder	Comprendre le cycle de vie d'un view controller	Récupérer des vraies données (format JSON)
Utiliser Swift : Tableaux et dictionnaires	Utiliser Auto Layout	Utiliser des classes Swift	
Utiliser Swift : Fonctions et closures	Manipuler les contrôles d'interface iOS	Utiliser le simulateur	

# OBJECTIFS / FINALITÉS



# OBJECTIFS / FINALITÉS

- Deezer : <http://api.deezer.com/search/artist?q=f>
- LaFourchette : [https://api.lafourchette.com/api?  
key=IPHONEPRODEDCRFV&method=restaurant get info&id restaurant=6861](https://api.lafourchette.com/api?key=IPHONEPRODEDCRFV&method=restaurant_get_info&id_restaurant=6861)  
[6861, 16409, 14163]
- Le Monde : [http://api-cdn.lemonde.fr/ws/8/mobile/www/ios-phone/en continu/  
index.json](http://api-cdn.lemonde.fr/ws/8/mobile/www/ios-phone/en_continu/index.json)
- L'Equipe : [http://app.francefootball.fr/json/les plus/plus.json](http://app.francefootball.fr/json/les_plus/plus.json)



Swift

# SWIFT

Lundi 2 juin 2014, lors la conférence WWDC 2014 (Apple Worldwide Developers Conference)



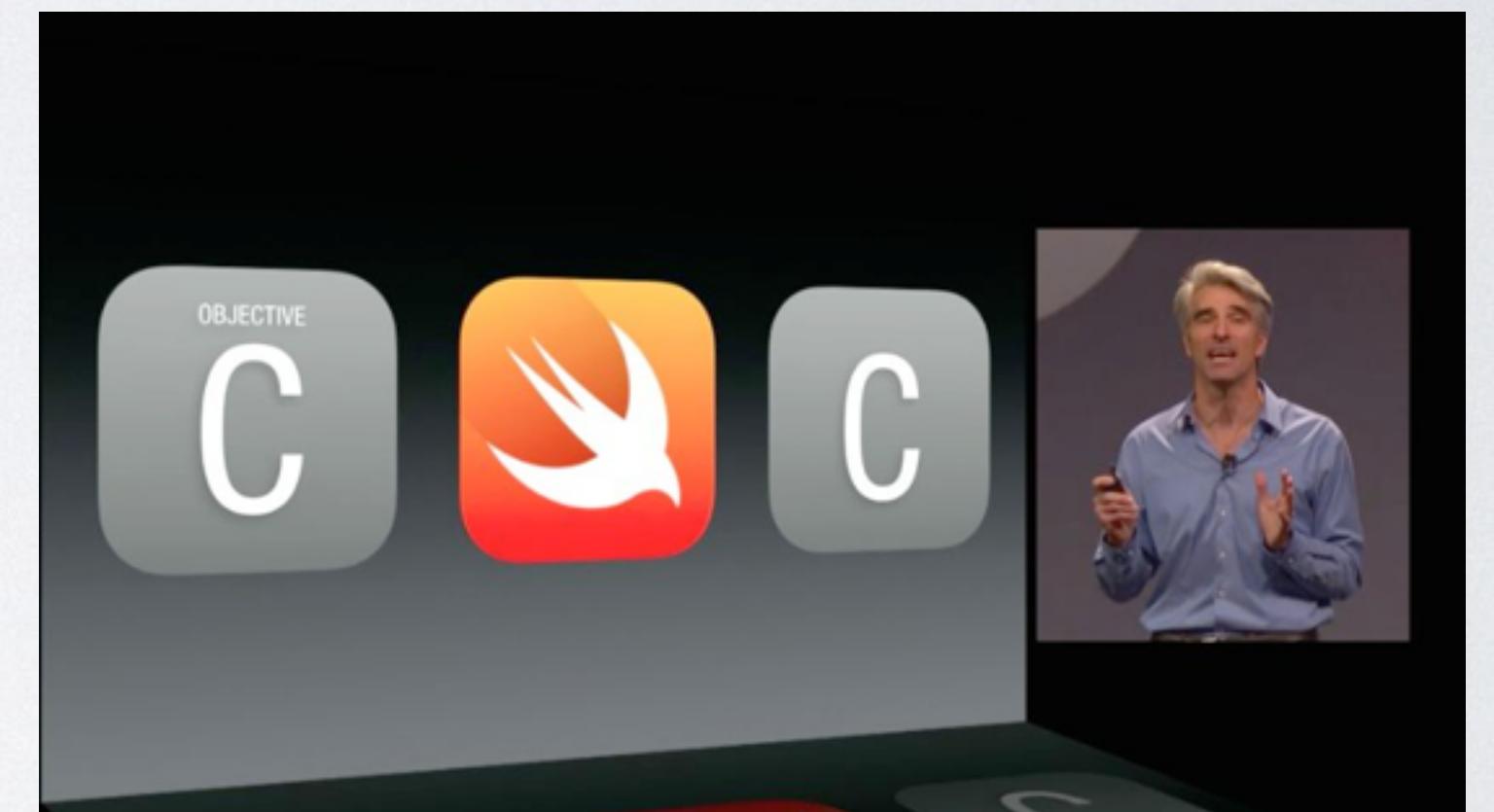
# SWIFT

Initié par Chris Lattner, informaticien américain chez Apple depuis 2005.  
Supervise le département Developer Tools.

Objective-C vs Swift (*« De l'Objective-C mais sans le C »*)

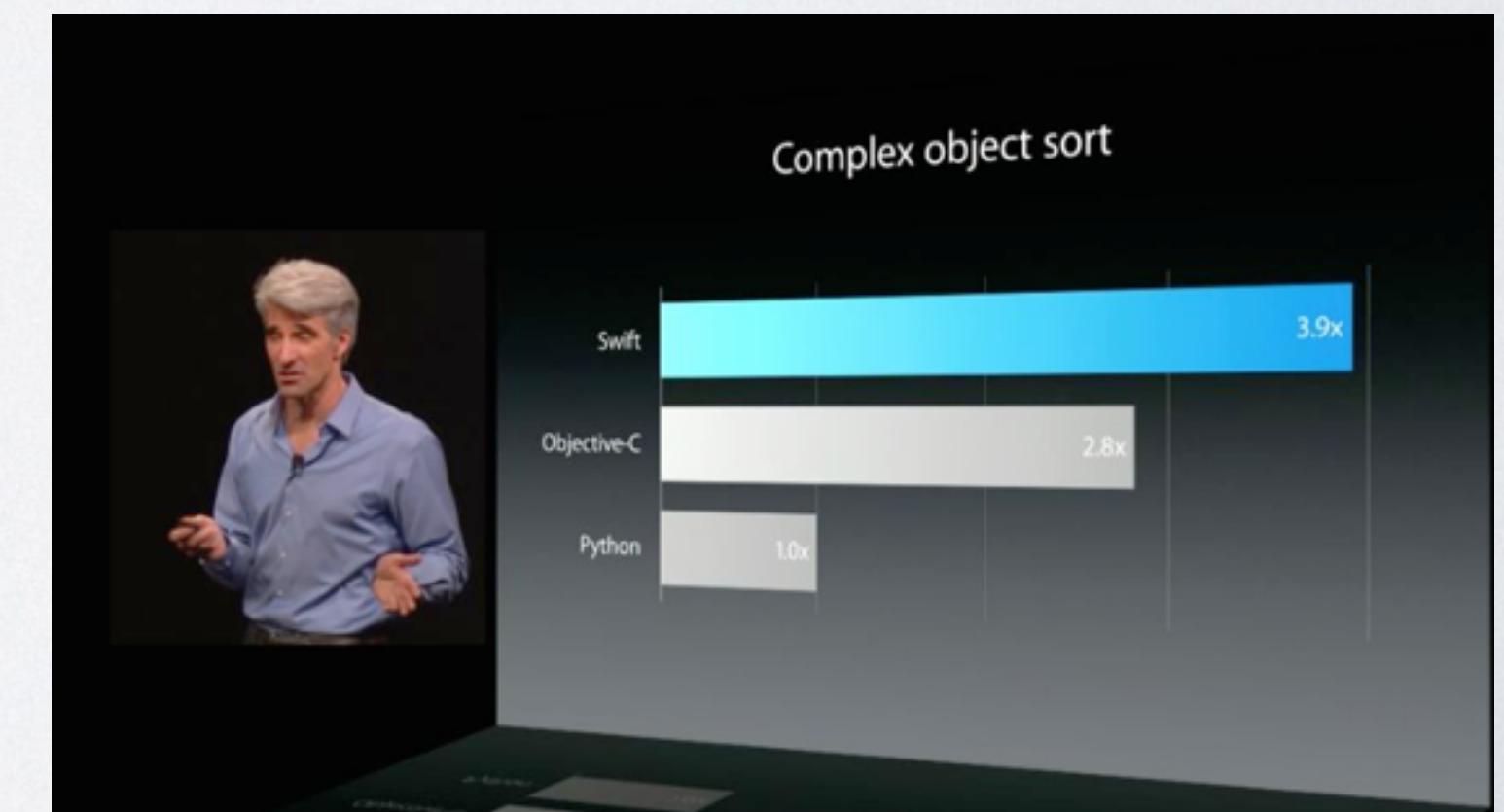
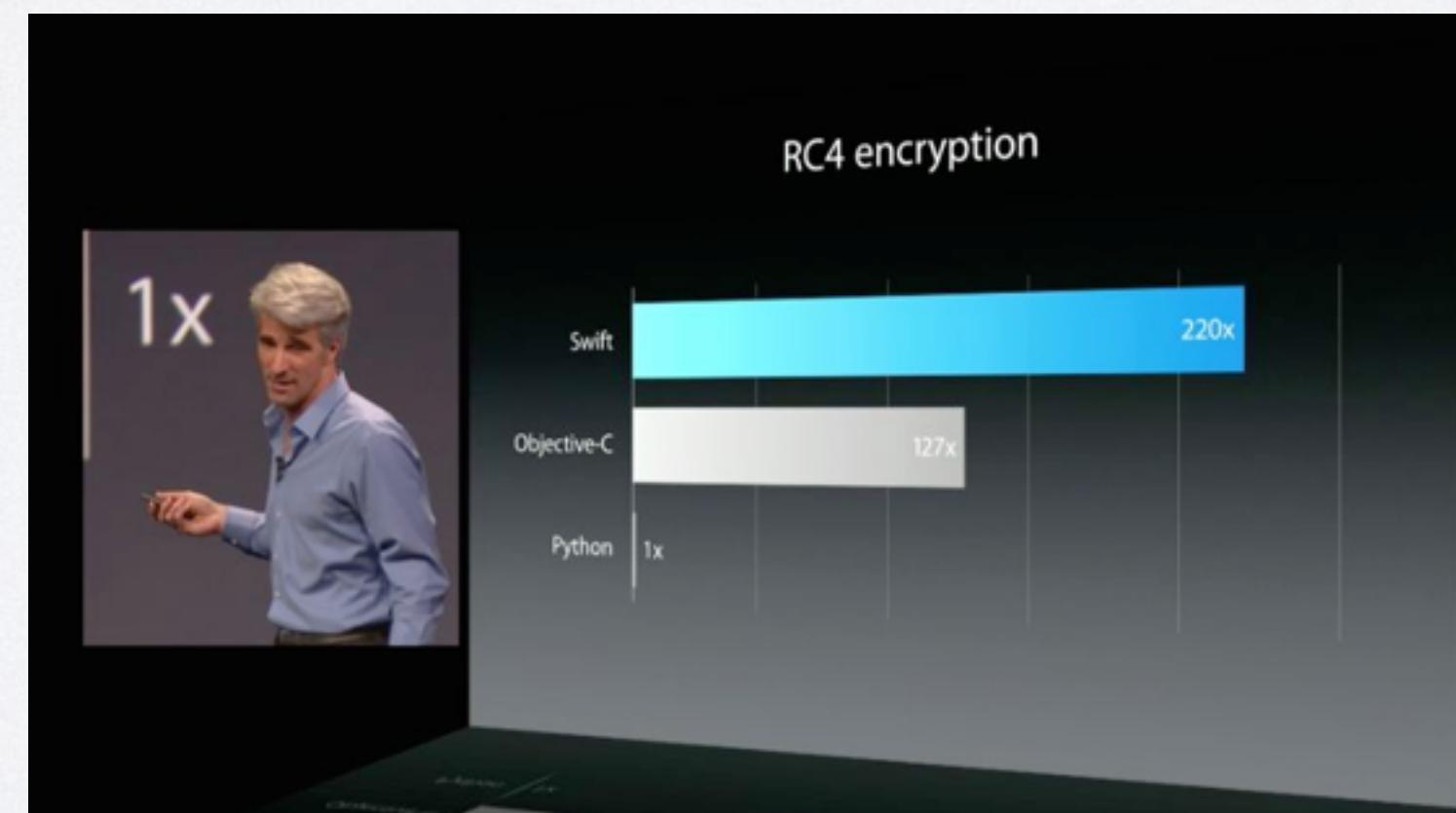
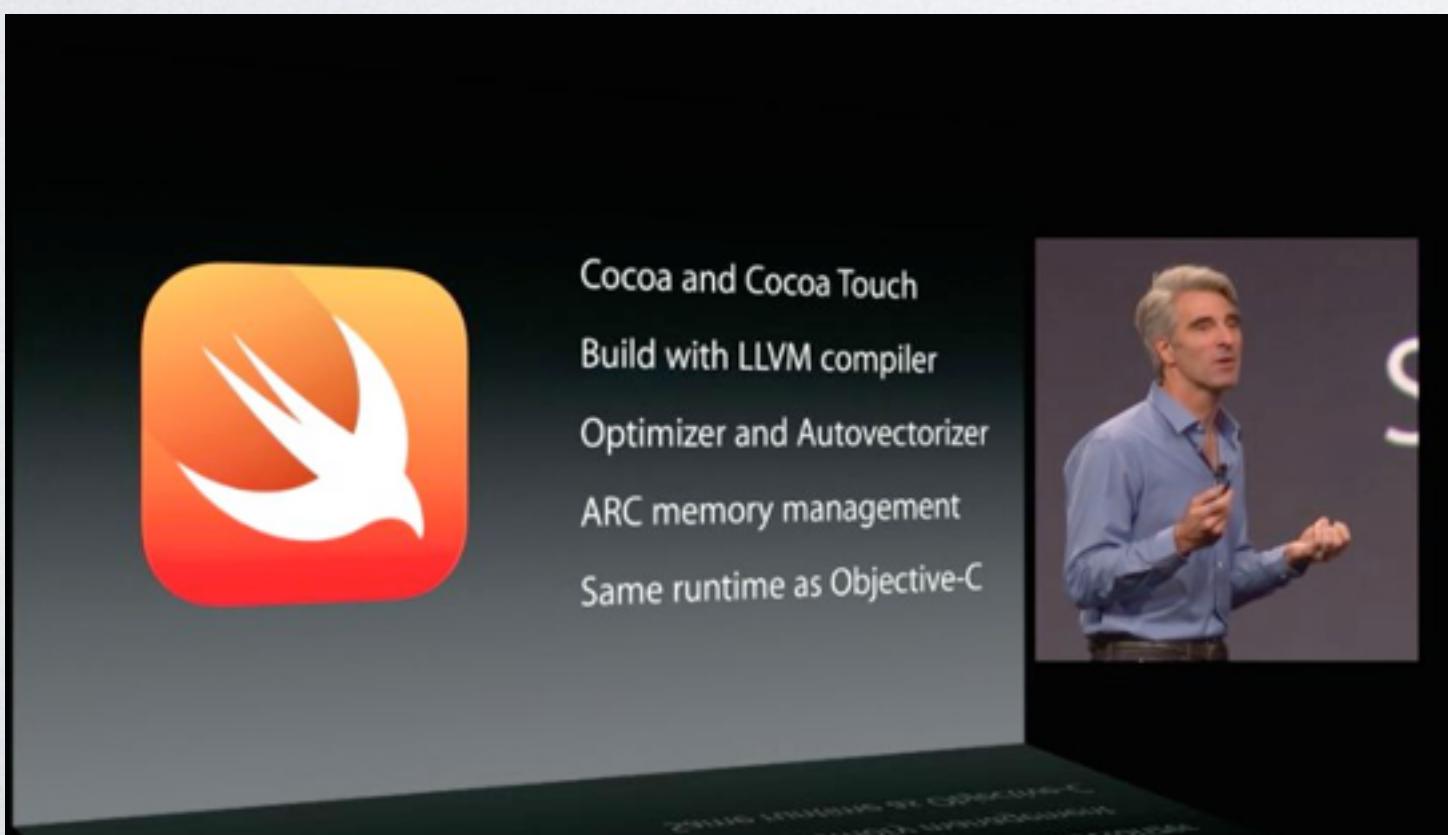
Version 4, Xcode9 et macOS10.12.6 minimum

Rapide, fonctionnel, performant et facile d'utilisation (selon Apple)



# SWIFT

Citation de Apple : « Swift est un nouveau langage de programmation à la fois puissant et intuitif, créé par Apple pour l'élaboration d'apps iOS et Mac. Il est conçu pour offrir aux développeurs toute la liberté et les capacités nécessaires pour produire la prochaine génération d'apps. Il ouvre à tous, développeurs ou non, tout un monde de possibilités. Swift est un langage facile à apprendre et à utiliser, même si vous n'avez jamais codé. Alors, si vous avez une idée d'app vraiment géniale, concrétisez-la avec Swift. »



# SWIFT

Variables, constantes et opérateurs

Conditions

Boucles

Tableaux et dictionnaires

Fonctions et closures

# SWIFT

Playground Xcode

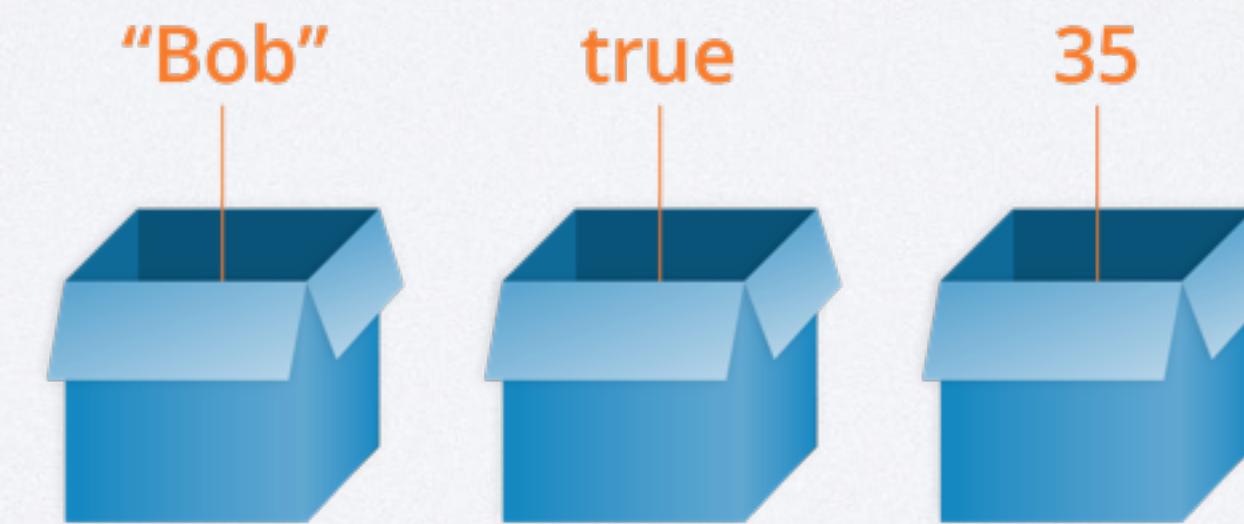


Contributions officielles Apple : <https://github.com/apple/swift>

De super exemples : <https://github.com/uraimo/Awesome-Swift-Playgrounds>

# SWIFT

Variables, constantes et opérateurs



# SWIFT

Variables, constantes et opérateurs

Fichiers .swift



**AppDelegate.swift**

Commentaires ???

# SWIFT

Variables, constantes et opérateurs

Commentaires

```
// Mon premier commentaire avec mon premier code Swift  
print("Hello, World!")
```

```
/*  
Mon second commentaire avec mon premier code Swift,  
mais sur plusieurs lignes  
 */  
print("Hello, World!")
```

# SWIFT

Variables, constantes et opérateurs

Variables

À quoi sert une variable ?

Déclaration ?

# SWIFT

Variables, constantes et opérateurs

Déclarations variables

```
var maVariable = 42  
var name = value
```

```
var maVariable = 42  
var mavariable = 42  
var maVariable = 42
```

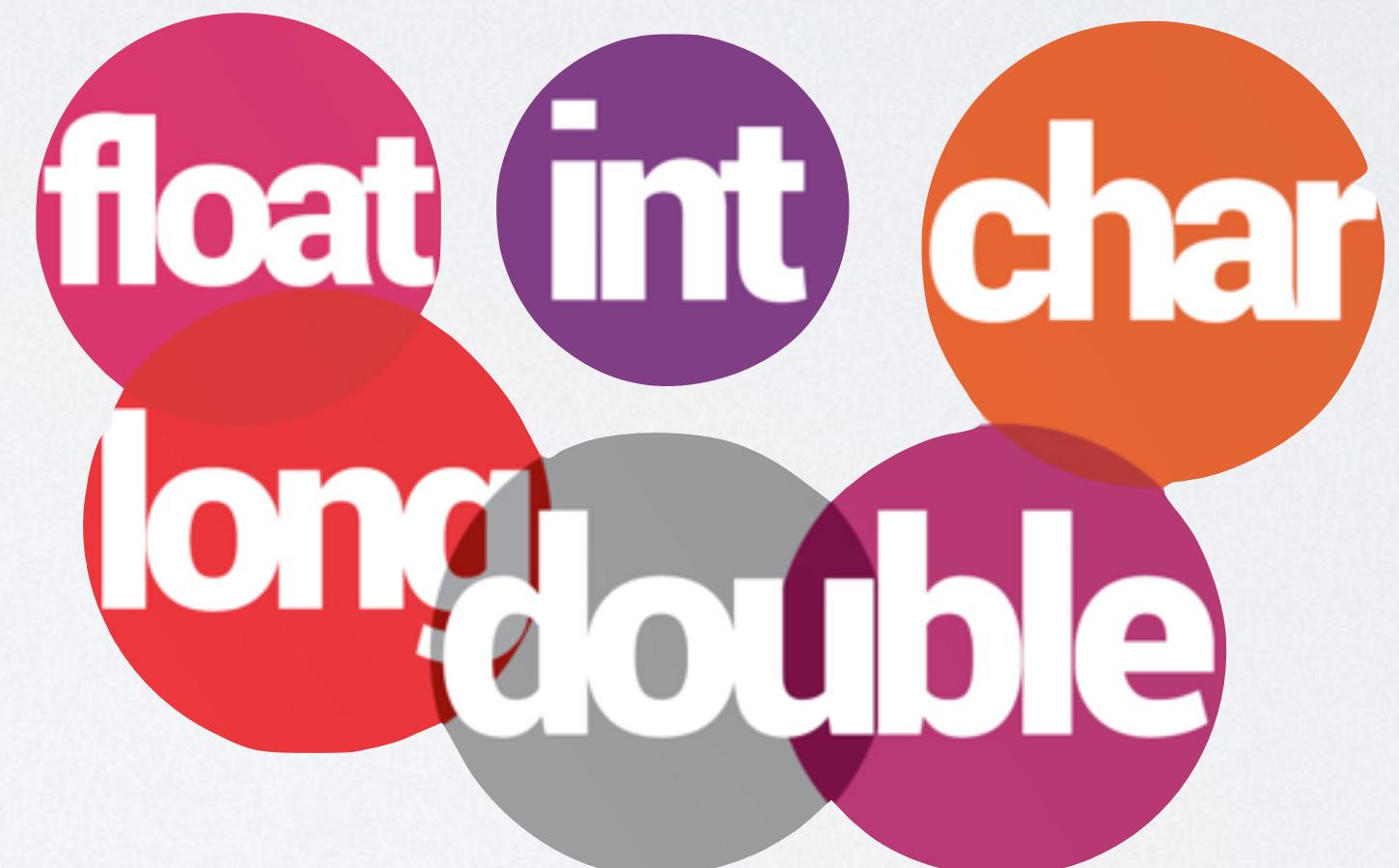
```
var myVariable = 42  
var mavariable = 42  
var maVariable = 42  
myVariable = 50  
mavariable = 72  
maVariable = 95
```

# SWIFT

Variables, constantes et opérateurs

Type de valeurs variables (entier, réel, chaînes de caractère, booléen)

```
var nombreEntier = 42  
var nombreReel = 3.1412  
var str = "Hello, playground!"  
var monBooleen = true  
var monBooleen = false
```



# SWIFT

Variables, constantes et opérateurs

Type implicite / explicite variables

```
var annee = 2018
```

```
var annee: Int
```

```
var annee: Int = 2018
print(annee)
```

```
var anneeBis: Int = 2018.5
print(annee)
```

```
Playground execution failed:
error: Formation.playground:4:21: error: cannot convert value of type 'Double' to specified type 'Int'
var anneeBis: Int = 2018.5
^~~~~~
Int( )
```

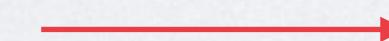
```
var annee: Int = 2018
var marque: String = "Audi"
var pi: Double = 3.1412
var avoirUneVoiture: Bool = false
```

# SWIFT

Variables, constantes et opérateurs

## Constantes

```
let pi: Double = 3.1412  
pi = 3.12
```



```
Formation.playground:1:1: note: change 'let' to 'var' to make it mutable  
let pi: Double = 3.1412  
^~~  
var
```

# SWIFT

Variables, constantes et opérateurs

Tuples

```
let joueur = ("Ronaldo", 33)
```

Type -> (String, Int)

```
let joueur = ("Ronaldo", 33)
print(joueur.0)

let (nom, age) = joueur
print("Nom : " + nom + " - " + "Age : \(\age)")

let (nom, _) = joueur
print("Nom : " + nom)

let joueur = (nom: "Ronaldo", age: 33)
print("Nom : " + joueur.nom + " - " + "Age : \(joueur.age)")
```

# SWIFT

Variables, constantes et opérateurs

## Opérateurs

Signe	Signification
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo

# SWIFT

Variables, constantes et opérateurs

## Opérateurs

```
let addition = 5 + 10
print(addition) // Résultat : 15

let soustraction = 20 - 7
print(soustraction) // Résultat : 13

let multiplication = 30 * 5
print(multiplication) // Résultat : 150

let division = 100 / 5
print(division) // Résultat : 20

let operation1 = (25 + 10) * 10
let operation2 = 25 + (10 * 10)
print(operation1) // Résultat : 350
print(operation2) // Résultat : 125

let nombre1 = 25
let nombre2 = 10
let operation3 = (nombre1 + nombre2) * nombre2
print(operation1) // Résultat : 350
```

```
let division = 100 / 11 // Dans 100 combien de fois 11 ?
let modulo = 100 % 11 // Quel est le reste de 100 / 11 ?

/*
Dans 100 il y a 9 fois 11
Il reste alors 1
*/
print(division) // Affiche 9
print(modulo) // Affiche 1
```

# SWIFT

Variables, constantes et opérateurs

Concaténations variables et Conversion type

```
let swift = "Swift,"  
let cool = " c'est trop cool !"  
let phraseComplete = swift + cool  
print(phraseComplete) // Résultat : Swift, c'est trop cool !  
print(swift + cool) // Résultat : Swift, c'est trop cool !  
  
let nombre1 = 10  
let nombre2 = 20  
let nombre = "\u{nombre1}" + "\u{nombre2}"  
print(nombre) // Résultat : 1020  
  
let nombre3 = 33  
print("Cristiano Ronaldo a \u{nombre3} ans") // Résultat : Cristiano Ronaldo a 33 ans  
print("Cristiano Ronaldo a " + String(nombre3) + " ans") // Résultat : Cristiano Ronaldo a 33 ans
```

# SWIFT

Variables, constantes et opérateurs

On résume ?

Variable = information en mémoire avec un nom et une valeur

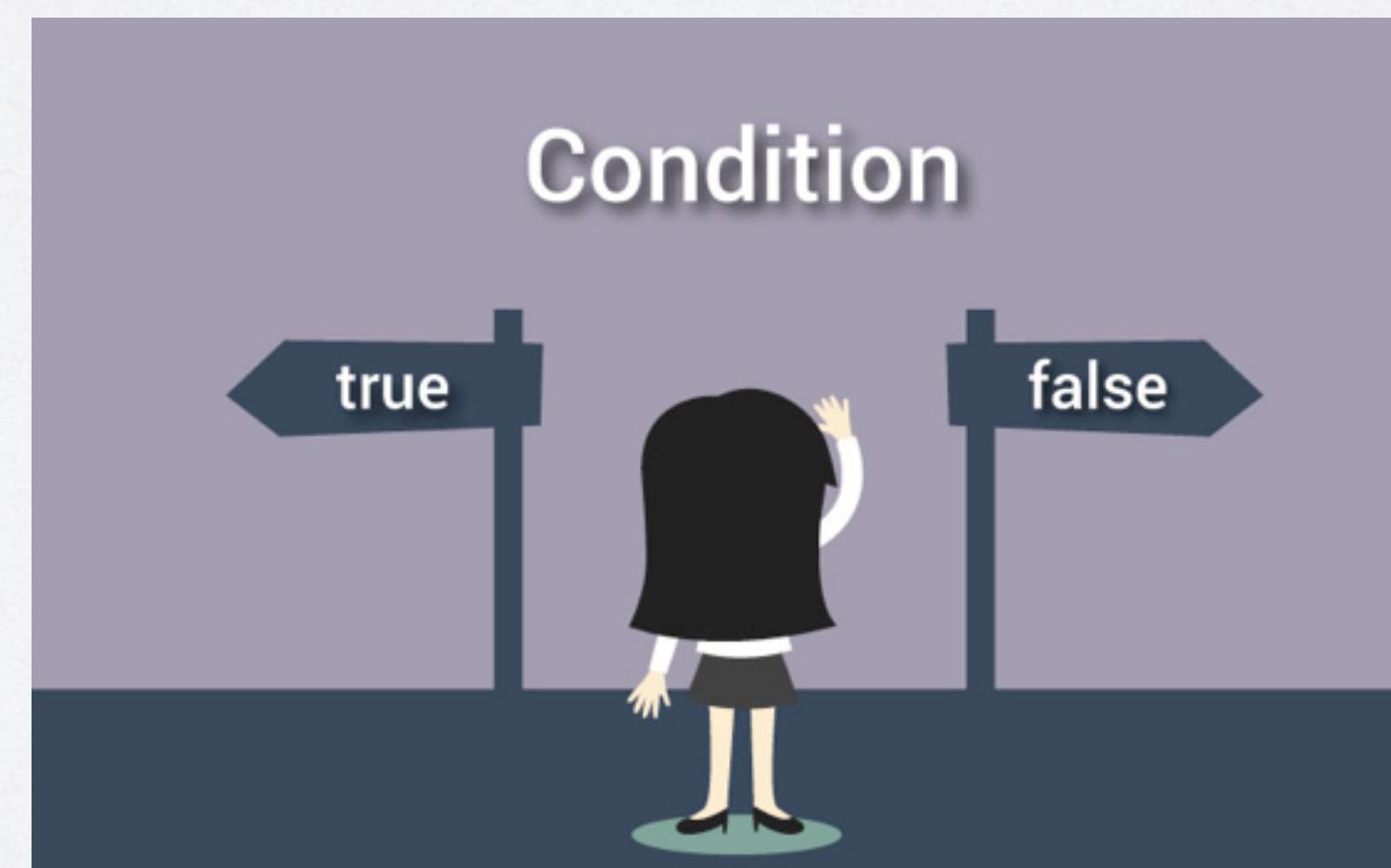
Types de valeurs variable : entier, décimal, chaîne de caractères, ...

Variable = « var » et Constante = « let »

Opérations possibles entre des variables : addition, soustraction, etc.

# SWIFT

## Conditions



# SWIFT

Conditions

Conditions If...Else, Else If ???

Opérations de comparaison

# SWIFT

Conditions

Opérateurs de comparaison

Symbol	Signification
==	est égale à
>	est supérieur à
>=	est supérieur ou égale à
<	est inférieur à
<=	est inférieur ou égale à
!=	est différent de

# SWIFT

## Conditions

If

```
let joueur = (nom: "Ronaldo", age: 33)
if joueur.age > 30 {
    print("Ce joueur est un trentenaire, bientôt la retraite dorée!!!")
}

if joueur.nom == "Ronaldo" {
    print("Voici un VRAI joueur de légende!!!")
```

# SWIFT

Conditions

If...Else

```
let joueur = (nom: "Ronaldo", age: 33)
if joueur.nom == "Ronaldo" {
    print("Voici un VRAI joueur de légende!!!")
}

if joueur.nom != "Ronaldo" {
    print("Ce joueur n'est pas une légende!!!")
```



```
let joueur = (nom: "Ronaldo", age: 33)
if joueur.nom == "Ronaldo" {
    print("Voici un VRAI joueur de légende!!!")
} else {
    print("Ce joueur n'est pas une légende!!!")
```

# SWIFT

## Conditions

### If...Else If...Else

```
let joueur = (nom: "Ronaldo", age: 33)
if joueur.age > 18 {
    print("Ce joueur est majeur.")
}
if joueur.age > 30 {
    print("Ce joueur est trentenaire.")
}
```



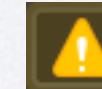
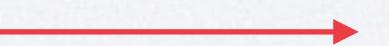
```
let joueur = (nom: "Ronaldo", age: 33)
if joueur.age > 18 {
    print("Ce joueur est majeur.")
} else if joueur.age > 30 {
    print("Ce joueur est trentenaire.")
} else {
    print("Ce joueur est mineur")
}
```

# SWIFT

Conditions

Conditions Multiples (Booléen)

```
if true {  
    print("Ce message s'affichera tout le temps.")  
}  
  
if false {  
    print("Ne s'exécutera jamais.")  
}
```



Will never be executed

# SWIFT

## Conditions

### Conditions Multiples (Booléen)

```
let booleen: Bool = true

if booleen == true {
    print("La variable booleen vaut vrai !")
} else {
    print("La variable booleen vaut faux !")
}

if booleen {
    print("La variable booleen vaut vrai !")
} else {
    print("La variable booleen vaut faux !")
}
```

```
let vrai: Bool = true
let faux: Bool = false

if vrai {
    print("Vrai vaut vrai")
}

if !faux {
    print("Un non faux est équivalent à vrai")
```

# SWIFT

## Conditions

### Des Conditions Multiples (Avec plusieurs conditions)

Signe	Signification	Vrai quand ?
&&	et	Vrai quand les deux valeurs sont vraies.
	ou	Vrai quand au moins une des deux valeurs est vraie.

# SWIFT

## Conditions

### Des Conditions Multiples (Avec plusieurs conditions)

```
let joueur = (nom: "Ronaldo", age: 33)

if (joueur.age == 33 && joueur.nom == "Ronaldo") || (joueur.age >= 18 && joueur.nom == "Messi") {
    print("Il ne peut y avoir qu'une super star sur le terrain!!!")
}
```

# SWIFT

## Conditions

## Switch

```
let noteJoueur = 10

if noteJoueur == 10 {
    print("Parfait")
} else if noteJoueur == 9 {
    print("Exceptionnel")
} else if noteJoueur == 8 {
    print("Très bon")
} else if noteJoueur == 7 {
    print("Bon")
} else if noteJoueur == 6 {
    print("Satisfaisant")
} else if noteJoueur == 5 {
    print("Moyen")
} else if noteJoueur == 4 {
    print("Insuffisant")
} else if noteJoueur == 3 {
    print("Mauvais")
} else if noteJoueur == 2 {
    print("Très mauvais")
} else if noteJoueur == 1 {
    print("Exécable")
} else {
    print("Désolé, il faut avoir une de ces notes pour être classé.")
}
```

# SWIFT

## Conditions

### Switch

```
let noteJoueur = 10

switch noteJoueur {
case 10:
    print("Parfait")
case 9:
    print("Exceptionnel")
case 8:
    print("Très bon")
case 7:
    print("Bon")
case 6:
    print("Satisfaisant")
case 5:
    print("Moyen")
case 4:
    print("Insuffisant")
case 3:
    print("Mauvais")
case 2:
    print("Très mauvais")
case 1:
    print("Exécable")
default:
    print("Désolé, il faut avoir une de ces notes pour être classé.")
}
```

# SWIFT

Conditions

Switch

```
let noteJoueur = 10

switch noteJoueur {
case 10, 9, 8, 7, 6:
    print("Plutôt positif")
case 5:
    print("Moyen")
case 4, 3, 2:
    print("Plutôt négatif")
case 1:
    print("Exécrable")
default:
    print("Désolé, il faut avoir une de ces notes pour être classé.")}
```

```
let noteJoueur = 8

switch noteJoueur {
case 1:
    print("Exécrable")
case 2...4:
    print("Plutôt négatif")
case 5:
    print("Moyen")
case 6...10:
    print("Plutôt positif")
default:
    print("Désolé, il faut avoir une de ces notes pour être classé.")}
```

# SWIFT

Conditions

Switch

```
let noteJoueur = 4

switch noteJoueur {
case 1:
    print("Exécable")
case 2..<4:
    print("Plutôt négatif")
case 4..<5:
    print("Moyen")
case 6...10:
    print("Plutôt positif")
default:
    print("Désolé, il faut avoir une de ces notes pour être classé.")
}
```

# SWIFT

Conditions

Conditions ternaires

```
let noteJoueur = 10
var meilleurJoueur: Bool

// If...Else classique
if noteJoueur == 10 {
    meilleurJoueur = true
} else {
    meilleurJoueur = false
}

// Condition Ternaire qui est identique
meilleurJoueur = noteJoueur == 10 ? true : false

// Identique également
meilleurJoueur = (noteJoueur == 10)
```

# SWIFT

Conditions

On résume ?

Conditions = effectuer des actions en fonctions de la valeur des variables

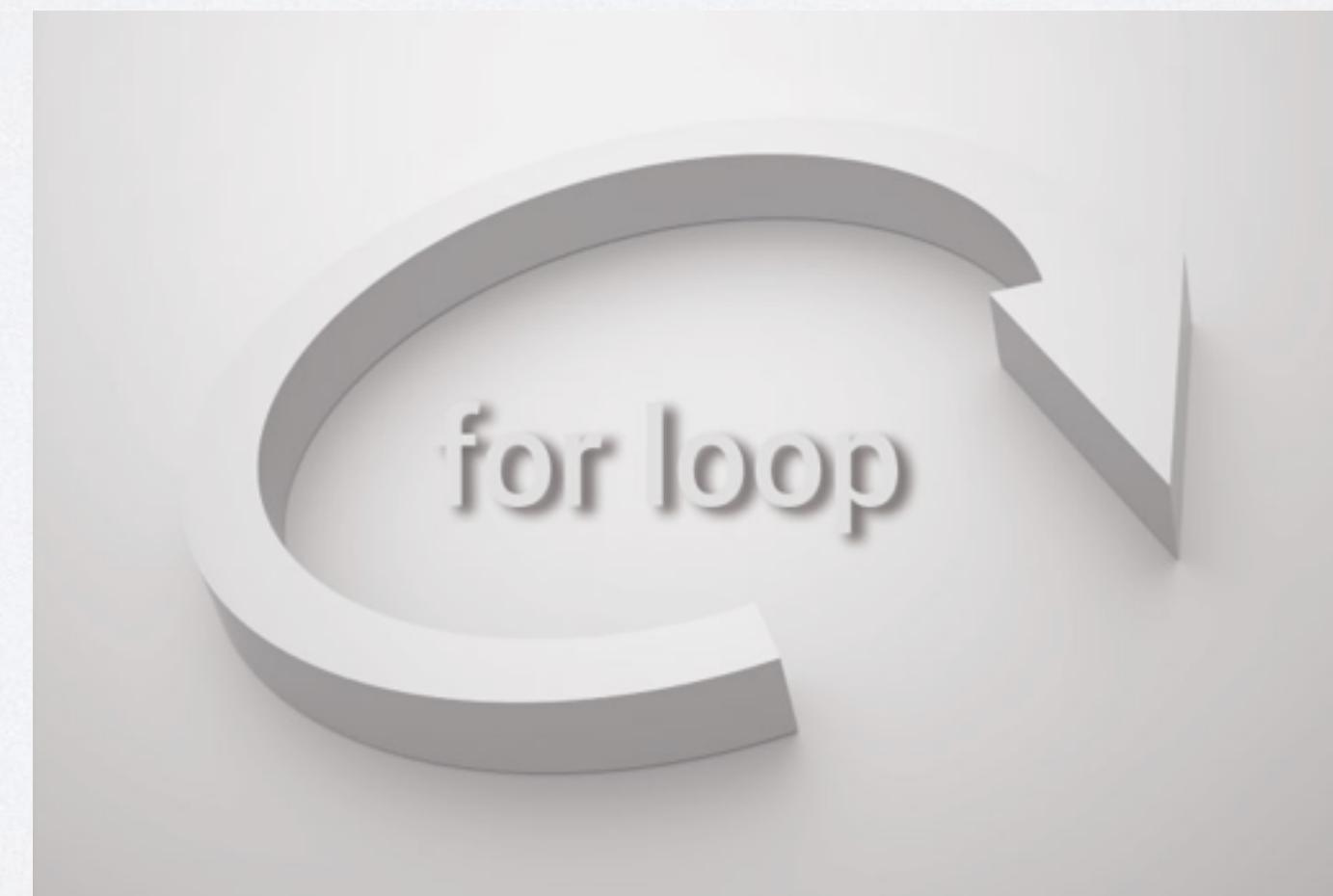
Types de conditions : if... else if... else, switch

Switch ne permet de tester que l'égalité

Combiner des conditions pour n'en former qu'une seule à l'aide des opérateurs && et ||

# SWIFT

Boucles

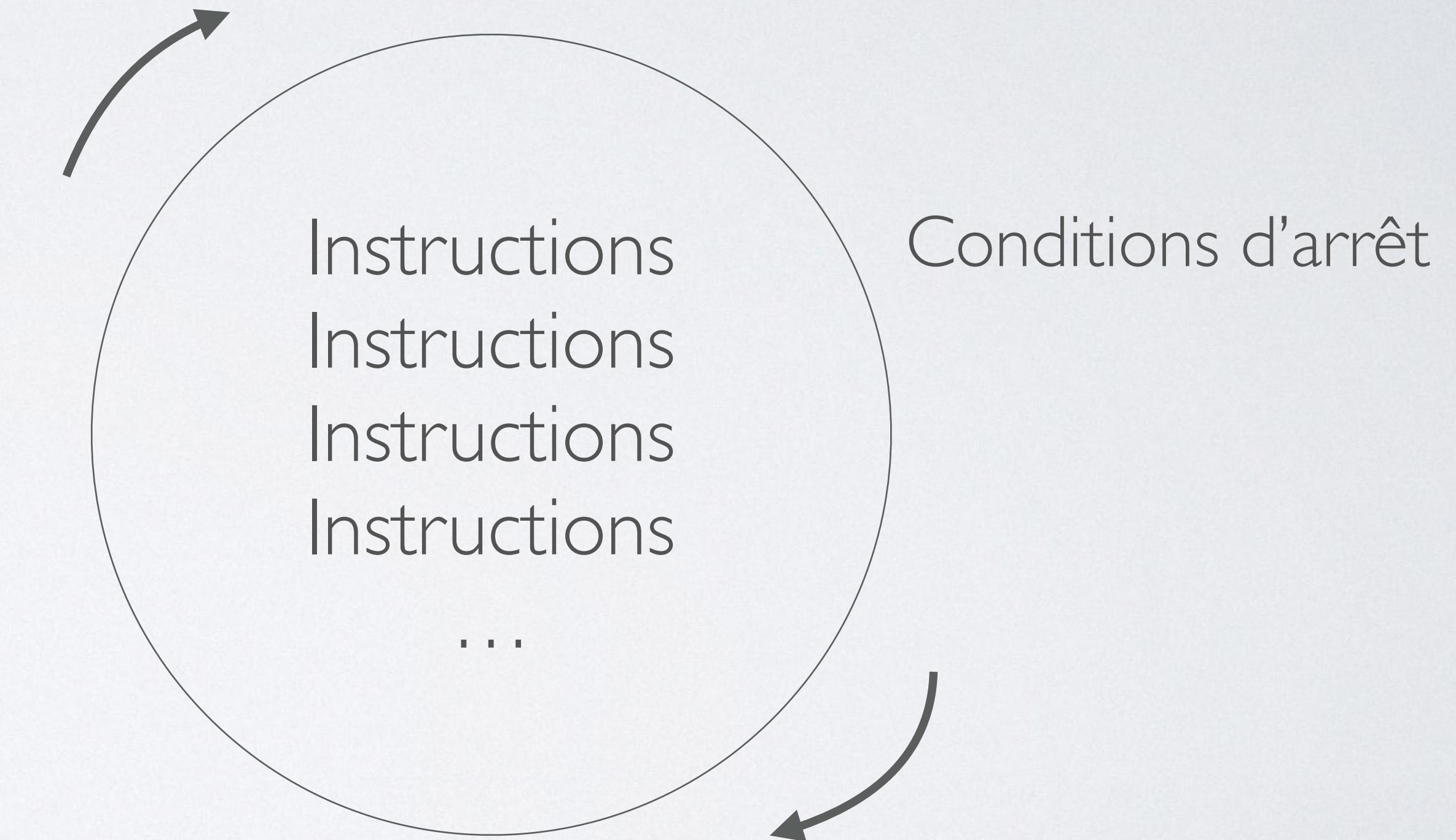


# SWIFT

Boucles

Boucles ???

Fonctionnement ???



# SWIFT

Boucles

While (Tant que)

```
while condition {  
    code  
}
```

```
var nombreAffichage: Int = 1  
  
while nombreAffichage <= 1000 {  
    print("\(joueur.nom) est une vrai star.")  
    nombreAffichage += 1 // Équivalent à : nombreAffichage = nombreAffichage + 1  
}
```

# SWIFT

Boucles

Repeat...While (Faire/Répéter...Tant que)

```
var nombreAffichage: Int = 1
repeat {
    print("\(joueur.nom) est une vrai star.")
    nombreAffichage += 1
} while nombreAffichage <= 1000
```

# SWIFT

Boucles

For-in

```
var nombreAffichage: Int  
for nombreAffichage in 1...1000 {  
    print("\(joueur.nom) est une vrai star à la puissance \(nombreAffichage)")  
}
```

```
for _ in 1..  
    print("\(joueur.nom) est une vrai star.")  
}
```

```
for item in shoppingList {  
    print(item)  
}  
for (index, value) in shoppingList.enumerated() {  
    print("Item \(index + 1): \(value)")  
}
```

Bonus avec un tableau (On verra par la suite...)

# SWIFT

Boucles

On résume ?

Boucles = réaliser une suite d'instructions plusieurs fois

Types de conditions : for (exécuter un nombre précis de fois une suite d'instruction)  
while / repeat ... while (s'exécute tant que la condition de la boucle n'est pas vérifié)

Dynamiser / Automatiser notre code

# SWIFT

Tableaux et Dictionnaires

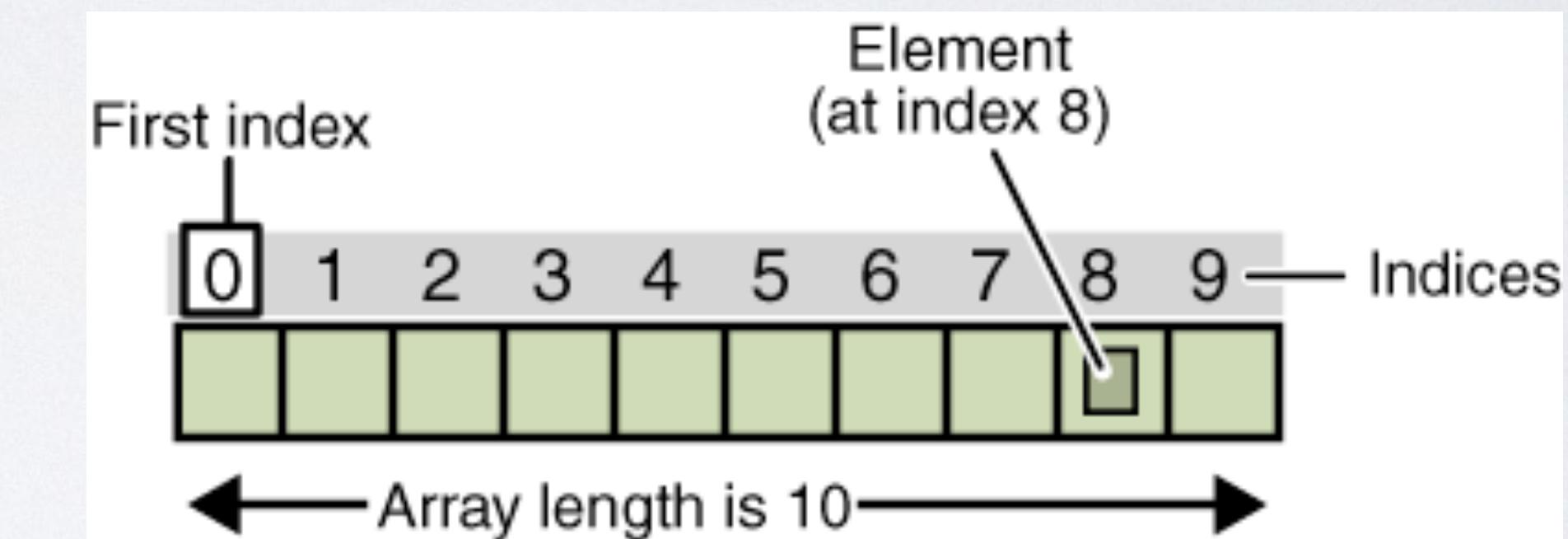


# SWIFT

Tableaux et Dictionnaires

Tableaux ???

Intérêt ???



# SWIFT

## Tableaux et Dictionnaires

Tableaux ???

Intérêt ???

```
let joueur1 = "Ronaldo"
let joueur2 = "Benzema"
let joueur3 = "Bale"
let joueur4 = "Kroos"
let joueur5 = "Modric"
```



```
let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]
```

# SWIFT

## Tableaux et Dictionnaires

Tableau = Peut contenir que des données du même type (Dans notre exemple que des « String »)

Tableau numéroté = Tableau avec des positions (Ex: Nom du joueur en position 1, position 2, ...)

Swift = Pas de « position », mais « Indice de tableau » avec comme démarrage la valeur 0

# SWIFT

Tableaux et Dictionnaires

Déclaration tableau

```
let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]
```

```
let joueurRealMadrid: [String]
```

# SWIFT

Tableaux et Dictionnaires

Accès élément tableau

```
let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]
print("Joueur 2 : " + joueurRealMadrid[1])
```

Test avec un index 6 ?

# SWIFT

## Tableaux et Dictionnaires

### Ajout / Modification valeur tableau

```
var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]

// Ajout en fin de tableau
joueurRealMadrid = joueurRealMadrid + ["Ramos"]
print(joueurRealMadrid)
// Ou comme de cette manière en concaténant
joueurRealMadrid += ["Carvajal"]
joueurRealMadrid += ["Navas", "Asencio", "Hernandez"]
print(joueurRealMadrid)

// Ajout au début du tableau
joueurRealMadrid = ["Isco"] + joueurRealMadrid
print(joueurRealMadrid)

// En utilisant des méthodes sur Array
joueurRealMadrid.append("Casemiro")
joueurRealMadrid.insert("Marcelo", at: 0)
print(joueurRealMadrid)
```

# SWIFT

## Tableaux et Dictionnaires

### Ajout / Modification valeur tableau

```
var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]

// Modification valeur éléments du tableau
joueurRealMadrid[0] = "Ceballos"
joueurRealMadrid[4...6] = ["Kovacic", "Varane"]
print(joueurRealMadrid)

// Suppression en utilisant des méthodes sur Array
joueurRealMadrid.remove(at: 0)
joueurRealMadrid.removeLast()
print(joueurRealMadrid)

// Connaitre le contenu en utilisant des méthodes sur Array
joueurRealMadrid.isEmpty
```

# SWIFT

Tableaux et Dictionnaires

Dictionnaire ???

Intérêt ???

## Dictionnaires

"name"	"Bruce Wayne"
"age"	45
"hero"	"Batman"
"alive"	true



# SWIFT

## Tableaux et Dictionnaires

Dictionnaire = ne seront plus numérotés automatiquement, mais manuellement

Dictionnaire = pas obligé de numéroter => Possibilité de mettre à la place des « Double » ou encore des « String »

Dictionnaire = Key -> Valeur

# SWIFT

Tableaux et Dictionnaires

Déclaration dictionnaire

```
let joueur = ["Nom": "Ronaldo", "Prénom": "Cristiano", "Club": "Real Madrid", "Age": "33"]
```

```
var joueurA: [String: String]
```

/!\ Une clé est unique.

# SWIFT

Tableaux et Dictionnaires

Accès élément dictionnaire

```
let joueur = ["Nom": "Ronaldo", "Prénom": "Cristiano", "Club": "Real Madrid", "Age": "33"]
print("Joueur : " + joueur["Nom"] !)
```

Test avec une clé « toto » ?

# SWIFT

## Tableaux et Dictionnaires

### Ajout / Modification valeur dictionnaire

```
var joueur = ["Nom": "Ronaldo", "Prénom": "Cristiano", "Club": "Real Madrid", "Age": "33"]

// Ajout valeur au dictionnaire
joueur["Pays"] = "Portugal"
print(joueur)

// Modification la valeur d'un élément du dictionnaire
joueur["Nom"] = "CR7"
print(joueur)

joueur.updateValue("34", forKey: "Age")
print(joueur)

// Suppression en utilisant des méthodes sur Dictionnaire
joueur.removeValue(forKey: "Club")
joueur["Nom"] = nil
print(joueur)

joueurA = [:]
print(joueur)

// Connaitre le contenu en utilisant des méthodes sur Dictionnaire
joueur.isEmpty
```

# SWIFT

## Tableaux et Dictionnaires

### Parcourir tableau / dictionnaire (Boucle For-in numéroté)

```
var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]
var i: Int

print("Les joueurs du Real Madrid sont : ")

for i in 0...<5 { // Indice du tableau débute à 0, Pourquoi ?
    print("*** " + joueurRealMadrid[i])
}
```

# SWIFT

## Tableaux et Dictionnaires

### Parcourir tableau (Boucle For-in non numéroté)

```
var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]
var i: Int

print("Les joueurs du Real Madrid sont : ")

for joueur in joueurRealMadrid {
    print("*** " + joueur)
}
```

En bonus si on veut l'index !!! →

```
var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]
var i: Int

print("Les joueurs du Real Madrid sont : ")

for (index, value) in joueurRealMadrid.enumerated() {
    print("Joueur \((index + 1): \(value))")
}
```

# SWIFT

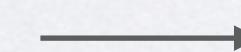
## Tableaux et Dictionnaires

### Parcourir dictionnaire (Boucle For-in non numéroté)

```
var joueur = ["Nom": "Ronaldo", "Prénom": "Cristiano", "Club": "Real Madrid", "Age": "33"]

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

En bonus si on  
veut la clé et  
valeur !!!



```
var joueur = ["Nom": "Ronaldo", "Prénom": "Cristiano", "Club": "Real Madrid", "Age": "33"]

for key in joueur.keys {
    print("Joueur key: \(key)")
}

for value in joueur.values {
    print("Joueur value: \(value)")
}

let joueurKeys = [String](joueur.keys)
let joueurValues = [String](joueur.values)

print("Joueur key: \(joueurKeys)")
print("Joueur value: \(joueurValues)")
```

# SWIFT

Tableaux et Dictionnaires

Recherche tableau

Petit exercice de recherche :

Essayer de rechercher « Messi » dans votre équipe du Real Madrid

# SWIFT

## Tableaux et Dictionnaires

### Recherche tableau : Idée de réponse

```
let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]
let joueurATrouver: String = "Messi"

// Indiquera si le joueur a été trouvé ou non
// Par défaut à false car on n'a pas encore commencé la recherche
var joueurTrouve: Bool = false

// On parcours notre équipe du Real Madrid
for joueur in joueurRealMadrid {

    // Si le joueur est dans notre équipe du Real Madrid,
    // alors on passe à true la variable
    if joueur == joueurATrouver {
        joueurTrouve = true
    }
}

if joueurTrouve {
    print(joueurATrouver + " est bien dans l'équipe du Real Madrid.")
} else {
    print("Jamais, Barcelona forever.")
}
```

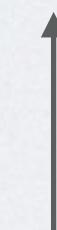
# SWIFT

## Tableaux et Dictionnaires

### Recherche tableau : Idée de réponse

```
let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]
let joueurATrouver: String = "Messi"

joueurRealMadrid.contains(joueurATrouver) ? print(joueurATrouver + " est bien dans l'équipe du Real Madrid.") : print("Jamais, Barcelona forever.")
```

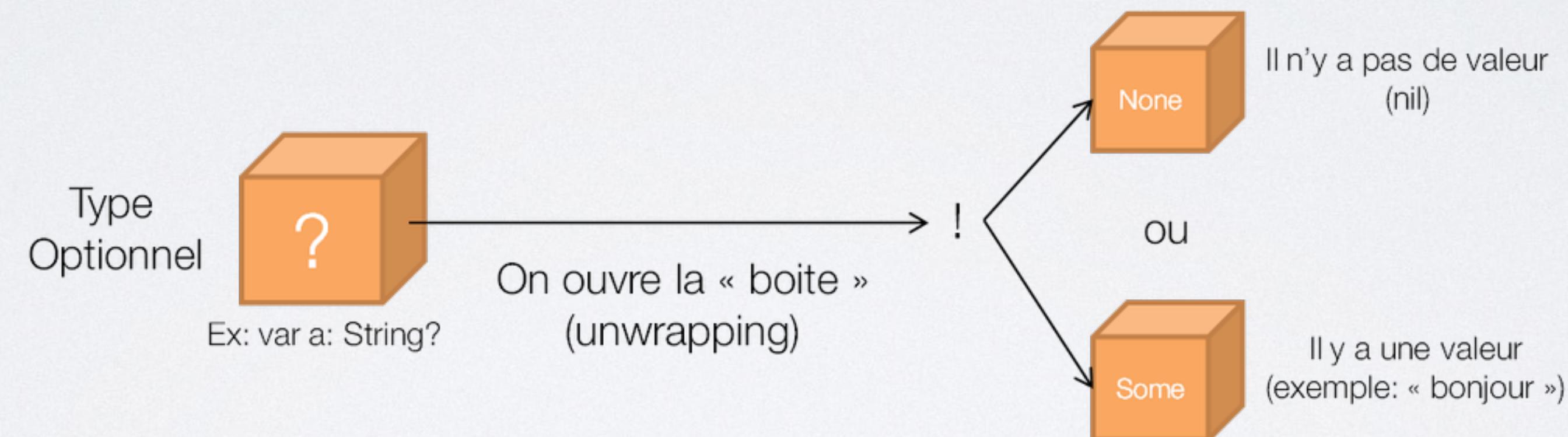


Astuce avec  
« contains »

# SWIFT

Tableaux et Dictionnaires

Type optionnel



# SWIFT

## Tableaux et Dictionnaires

## Type optionnel

```
var joueur = ["Nom": "Ronaldo", "Prénom": "Cristiano", "Club": "Real Madrid", "Age": "33"]

// Récupère le pays du joueur
let paysJoueur: String? = joueur["Pays"]

// Si la "boîte" contient une valeur
if paysJoueur != nil {
    // On "unwrap" avec "!" pour "ouvrir" la boîte
    print("Le pays du joueur est : " + paysJoueur!)
} else {
    print("Le pays du joueur n'est pas défini")
}
```

```
let joueurRealMadrid: [String?] = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]

if let joueur1 = joueurRealMadrid[0]?.uppercased(),
   let joueur2 = joueurRealMadrid[1]?.uppercased()  {
    print("\(\joueur1)", "\(\joueur2)")
} else {
    print("Aucun joueur en 1ère et 2ème position")
}
```

# SWIFT

Tableaux et Dictionnaires

## Quelques Méthodes utiles

Map = Fonction qui boucle sur chaque élément de la collection et lui applique une opération

Reduce = Fonction qui boucle sur chaque élément de la collection et les combine en une seule valeur

Filter = Fonction qui boucle sur chaque élément de la collection et retourne une nouvelle collection qui satisfait une condition d'inclusion

Sort = Fonction qui tri une collection selon un prédictat

# SWIFT

## Tableaux et Dictionnaires

### Quelques Méthodes utiles

```
// Map
let celcius = [-5.0, 10.0, 21.0, 33.0, 50.0]
let fahrenheit = celcius.map { $0 * (9/5) + 32 }
print(fahrenheit) // Output: [23.0, 50.0, 69.8, 91.4, 122.0]

// Reduce
let values = [3, 4, 5]
let sum = values.reduce(0, +)
print(sum) // Output: 12

// Filter
let numbers = [11, 13, 14, 17, 21, 33, 22]
let pair = numbers.filter { $0 % 2 == 0 }
print(pair) // Output: [14, 22]

// Chaining Map, Reduce, Filter
let years = [1989, 1992, 2003, 1970, 2014, 2001, 2015, 1990, 2000, 1999]
let sum1 = years.filter({ $0 >= 2000 }).map({ 2017 - $0 }).reduce(0, +)
print(sum1) // Output: 52

let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]

print(joueurRealMadrid.sorted(by: { $0 < $1 })) // En utilisant notre tri
print(joueurRealMadrid.sorted(by: <)) // En utilisant un opérateur
```

# SWIFT

Tableaux et Dictionnaires

On résume ?

Tableaux = variables avec plusieurs données du même type numérotées (indices) =>  
L'indice d'un tableau commence toujours à 0.

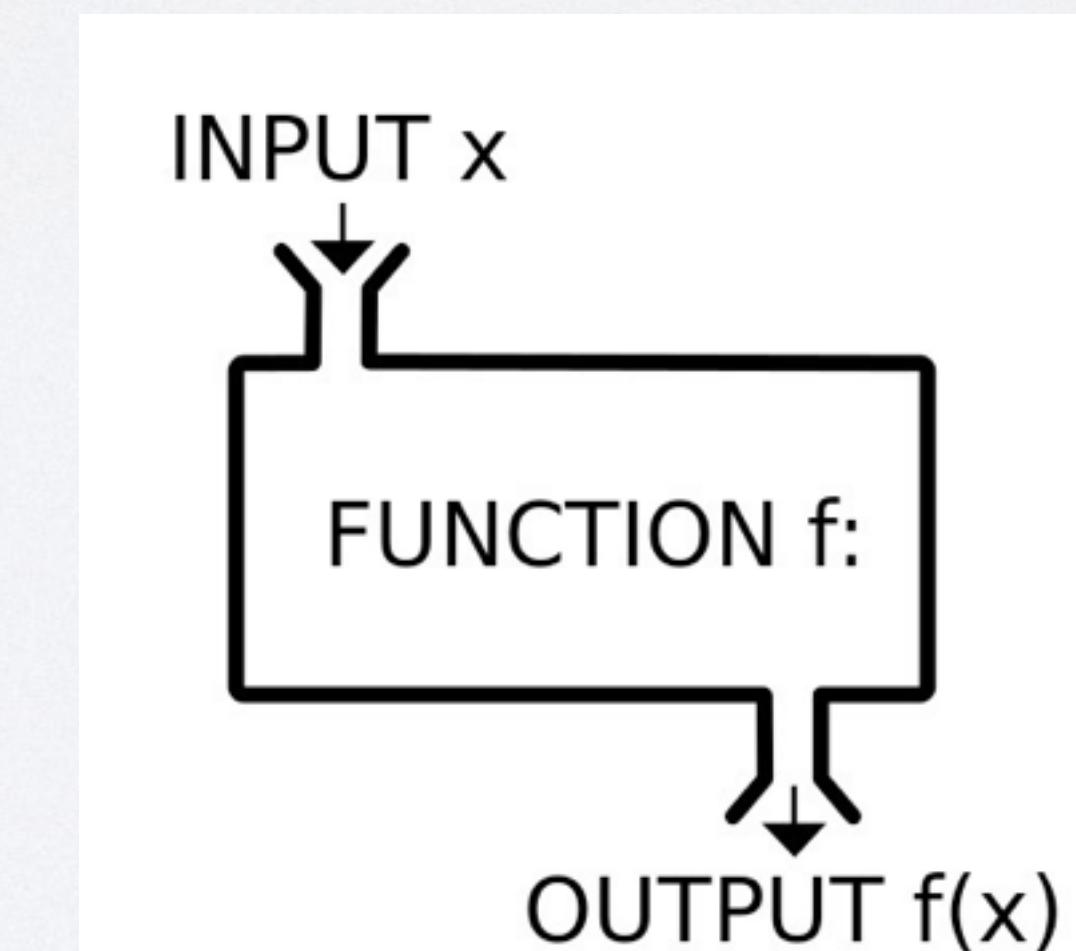
Dictionnaires = variables permettant d'associer un élément à un autre => (clé-valeur) : clés  
du même type et valeurs du même type, mais une clé peut être du même type qu'une  
valeur.

Déclaration tableau = [valeur1, valeur2, ...]

For-in = boucle la plus utile et utilisée

# SWIFT

## Fonctions et closures

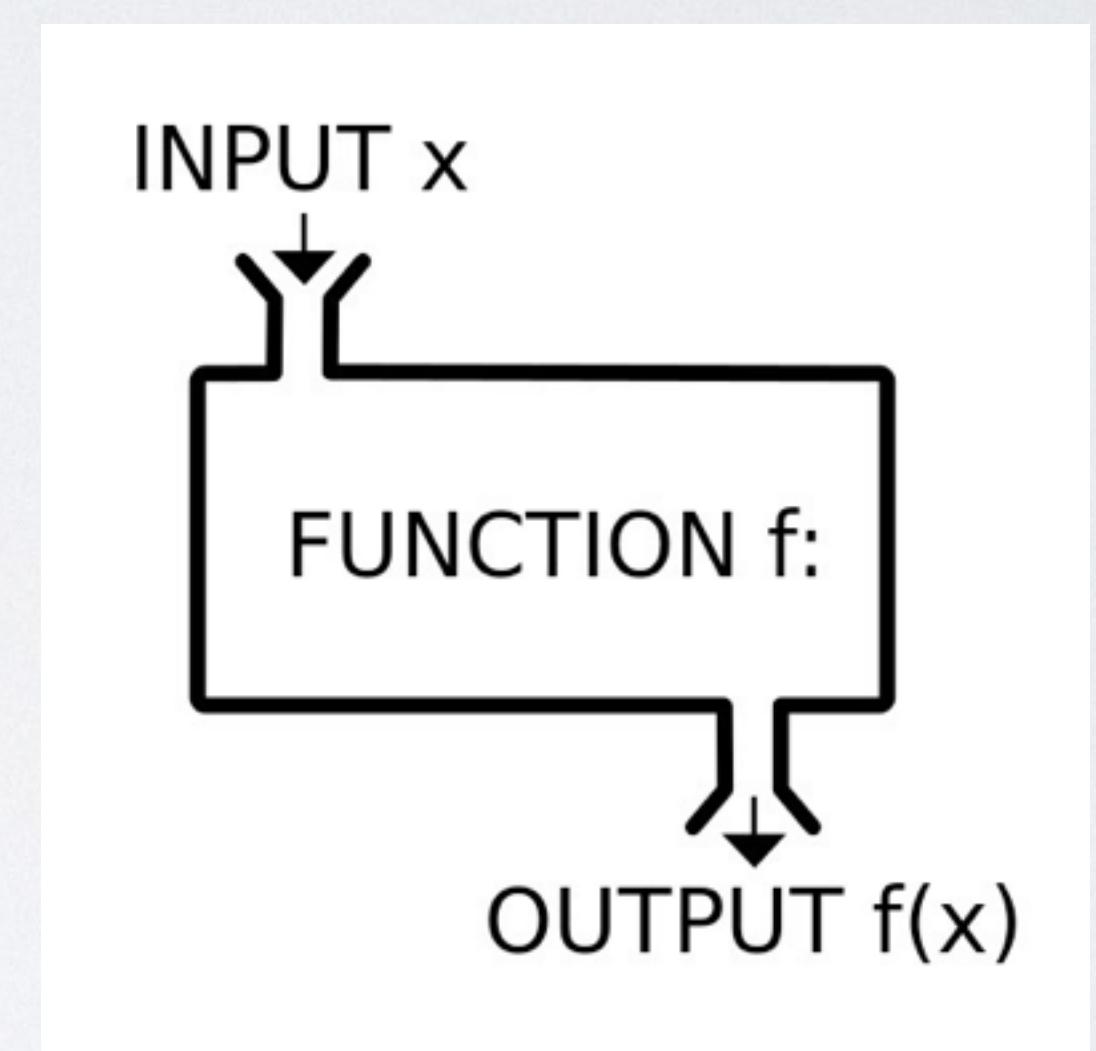


# SWIFT

Fonctions et closures

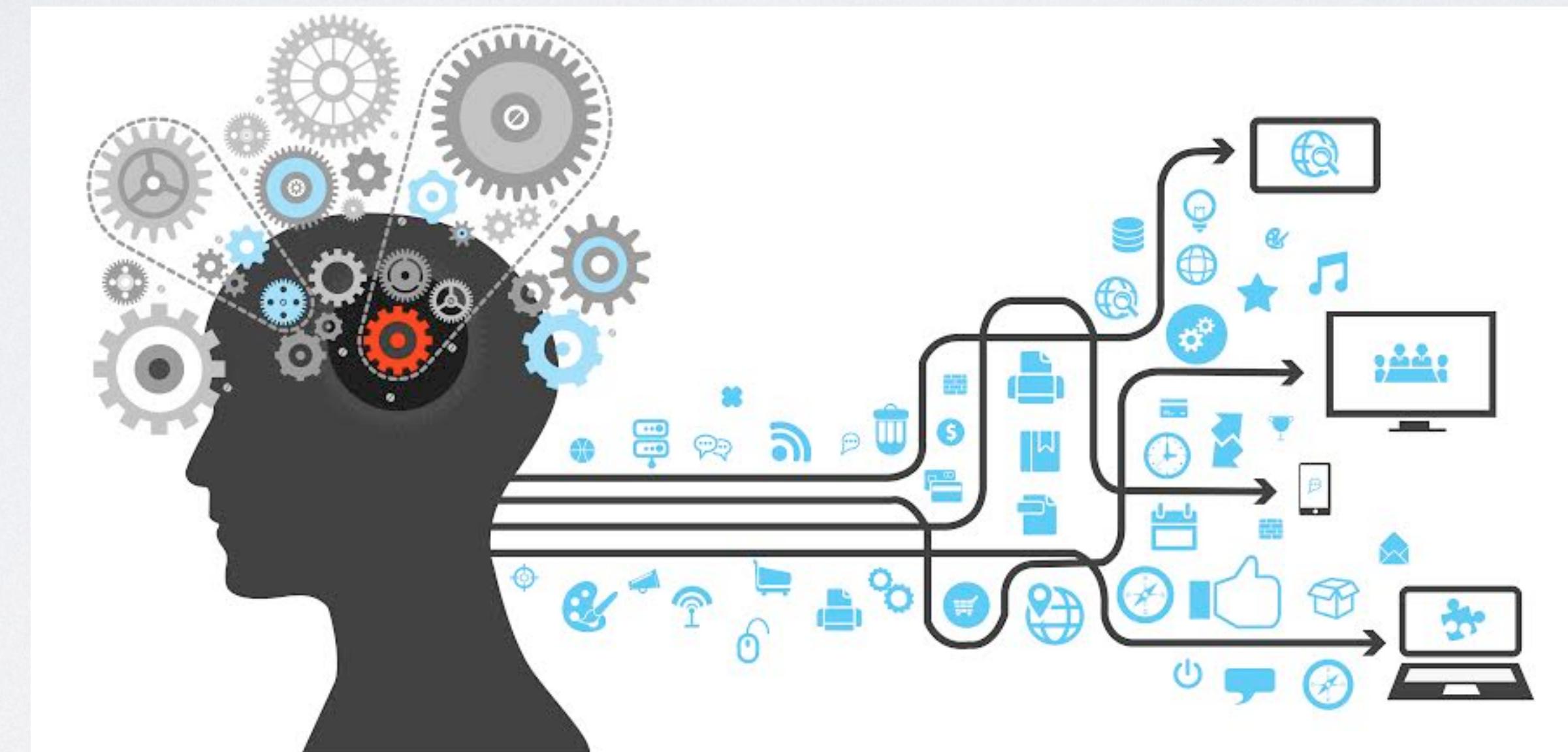
Fonction ???

Intérêt ???



# SWIFT

Fonctions et closures



# SWIFT

Fonctions et closures

Fonction sans paramètres

```
func name(parameters) -> return type {  
    function body  
}
```

```
let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]  
  
// Déclaration de la méthode  
func afficheLePremierJoueur() {  
    print(joueurRealMadrid[0])  
}  
  
// Appel de la méthode  
afficheLePremierJoueur()
```

# SWIFT

## Fonctions et closures

### Fonction sans paramètres

```
let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]

// Déclaration de la méthode
func afficheLePremierJoueur() {
    print(joueurRealMadrid[0])
}

// Déclaration de la seconde méthode
func afficheLeSecondJoueur() {
    afficheLePremierJoueur() // On appelle la 1ere méthode dans la 2nd
    print(joueurRealMadrid[1])
}

// Appel de la méthode
afficheLeSecondJoueur()
```

# SWIFT

## Fonctions et closures

### Fonction avec paramètres

```
var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]

func ajouteJoueur(joueur: String) {
    joueurRealMadrid.append(joueur)
    print("Nouveau joueur " + joueur + " ajouté")
    print("Nouvel effectif \(joueurRealMadrid)")
}

// Appel de la méthode
ajouteJoueur(joueur: "Isco")
```

```
func name(parameters) -> return type {
    function body
}
```

```
var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]

func ajouteJoueurs(joueur1: String, joueur2: String) {
    joueurRealMadrid.append(joueur1)
    joueurRealMadrid.append(joueur2)
    print("Nouveau joueur " + joueur1 + " ajouté")
    print("Nouveau joueur " + joueur2 + " ajouté")
    print("Nouvel effectif \(joueurRealMadrid)")
}

// Appel de la méthode
ajouteJoueurs(joueur1: "Isco", joueur2: "Varane")
```

# SWIFT

## Fonctions et closures

## Fonction avec paramètres

```
External   Internal
func display(Name name: String, Age age:Int,Address address: String)->Void{
    println(name+"\\(age)" +address)
}

External
|
display(Name: "Hafiz Waleed Hussain", Age: 25, Address: "Pakistan")
```

```
var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]

func ajouteJoueurs(AvecNom joueur1: String, AvecNom joueur2: String) {
    joueurRealMadrid.append(joueur1)
    joueurRealMadrid.append(joueur2)
    print("Nouveau joueur " + joueur1 + " ajouté")
    print("Nouveau joueur " + joueur2 + " ajouté")
    print("Nouvel effectif \(joueurRealMadrid)")
}

// Appel de la méthode
ajouteJoueurs(AvecNom: "Isco", AvecNom: "Varane")

var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]

func ajouteJoueurs(_ joueur1: String, AvecNom joueur2: String) {
    joueurRealMadrid.append(joueur1)
    joueurRealMadrid.append(joueur2)
    print("Nouveau joueur " + joueur1 + " ajouté")
    print("Nouveau joueur " + joueur2 + " ajouté")
    print("Nouvel effectif \(joueurRealMadrid)")
}

// Appel de la méthode
ajouteJoueurs("Isco", AvecNom: "Varane")
```

# SWIFT

Fonctions et closures

Fonction avec retours

```
func name(parameters) -> return type {  
    function body  
    return return value  
}
```

```
var joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]  
  
func presentDansEquipe(joueurATrouver joueur:String) -> Bool {  
    return joueurRealMadrid.contains(joueurATrouver)  
}  
  
// Appel de la méthode et récupération valeur  
let estPresent = presentDansEquipe(joueurATrouver: "Messi")  
estPresent ? print(joueurATrouver + " est bien dans l'équipe du Real Madrid.") : print("Jamais, Barcelona forever.")
```

# SWIFT

Fonctions et closures

Fonction comme paramètre ou type de retour

```
func fonction(fonctionParametre: (String, String) -> Bool) {  
    // Utilisation de la fonction passé en paramètre  
    fonctionParametre("Messi", "Mbappe")  
}
```

# SWIFT

## Fonctions et closures

### Fonction comme paramètre ou type de retour

```
func presentDansEquipe(joueur1: String, joueur2: String) -> Bool {
    let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]
    return (joueurRealMadrid.contains(joueur1) && joueurRealMadrid.contains(joueur2)) // retourne vrai si les 2 joueurs sont
dans l'équipe, sinon non
}

func validationEquipe(fonctionParametre: (String, String) -> Bool) {

    if fonctionParametre("Ronaldo", "Benzema") {
        print("Ces joueurs jouent bien dans cette équipe.")
    } else {
        print("Aucun de ces joueurs ne jouent dans cette équipe.")
    }
}

validationEquipe(fonctionParametre: presentDansEquipe)
```

# SWIFT

Fonctions et closures

Fonction comme paramètre ou type de retour

```
func validationEquipe(joueurs: [String]) -> (String) -> Bool {  
    // Nested Method  
    func presentDansEquipe(joueurATrouver: String) -> Bool {  
        return joueurs.contains(joueurATrouver)  
    }  
  
    return presentDansEquipe  
}  
  
let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]  
let estPresent = validationEquipe(joueurs: joueurRealMadrid)("Messi")  
  
estPresent ? print(joueurATrouver + " est bien dans l'équipe du Real Madrid.") : print("Jamais, Barcelona forever.")
```

# SWIFT

Fonctions et closures

## Closures

```
{ (parameters) -> return type in  
    statements  
}
```

```
let exempleClosure = {  
    print("Un exemple de super closure")  
}  
  
exempleClosure() // Le code est exécuté ici
```

# SWIFT

## Fonctions et closures

## Closures

```
func validationEquipe(fonctionParametre: (String, String) -> Bool) {  
  
    if fonctionParametre("Ronaldo", "Benzema") {  
        print("Ces joueurs jouent bien dans cette équipe.")  
    } else {  
        print("Aucun de ces joueurs ne jouent dans cette équipe.")  
    }  
}  
  
validationEquipe(fonctionParametre: { (joueur1: String, joueur2: String) -> Bool in  
    let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]  
  
    // retourne vrai si les 2 joueurs sont dans l'équipe, sinon non  
    return (joueurRealMadrid.contains(joueur1) && joueurRealMadrid.contains(joueur2))  
})
```

```
validationEquipe { (joueur1: String, joueur2: String) -> Bool in  
    let joueurRealMadrid = ["Ronaldo", "Benzema", "Bale", "Kroos", "Modric"]  
  
    // retourne vrai si les 2 joueurs sont dans l'équipe, sinon non  
    return (joueurRealMadrid.contains(joueur1) && joueurRealMadrid.contains(joueur2))  
}
```

# SWIFT

Fonctions et closures

On résume ?

Fonction = bout de code qui va permettre d'exécuter des instructions selon des paramètres fournis ou non avec valeur de retour ou non

Fonction démarre avec le mot clé « Func »

Fonction = Si valeur de retour alors « -> » et « return »

Closure = Sorte de fonction qui peuvent être définies directement lors de l'appel

# SWIFT

Un peu de pratique

Exemples exercices au choix

Réaliser une calculatrice (2 chiffres + 1 opérateur = résultat)

Réaliser un affichage de la table de multiplication

Réaliser une gestion d'équipe (effectif, but, statistique, ...)

# SWIFT

Pour aller plus loin



SWIFT

# SWIFT

Pour aller plus loin

<https://developer.apple.com/swift/>

[https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/](https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/)

<https://www.raywenderlich.com/category/swift>

<https://github.com/uraimo/Awesome-Swift-Playgrounds>

<http://swift-tuto.fr/apprendre-langage-swift/>



# XCODE

IDE (Integrated Development Environment) = Environnement de développement

Possibilité de développer en C, C++, Obj-C, Java, AppleScript, Python, Ruby, Rez et Swift.

Swift permet de développer des apps que pour les appareils iOS 7 ou supérieur



# XCODE

Utilisons XCode



# XCODE

Utilisons XCode

A quoi correspond ?

- Fenêtre, Plist, AppDelegate (Cycle de vie App), Run App, Simulateur, Interface Builder, ... ?
- UIView, UIViewController (Cycle de vie), Classe Control UI, Storyboard, Xibs, ... ?
- Breakpoint, Debug, Console, ... ?

# XCODE

Utilisons XCode

A quoi correspond ?

- Interaction élément code / UI (Delegate, IBOutlet, IBAction, ...) ?
- Héritage des vues, gesture recognizer, ... ?

# XCODE

Utilisons XCode

A quoi correspond ?

- Navigation, transition segue, modal/push controller, ... ?
- TableView, MVC, ... ?

# XCODE

Utilisons XCode

Exemples exercices au choix

- Réaliser un affichage de la table de multiplication (Tap ×10 sur Button -> Affichage Labels, ...)
- Afficher Tableau statique + Clic détail (Donnée détaillée, Webview, ...)

# IOS

Pour aller plus loin

<https://www.raywenderlich.com/category/ios>

<http://nshipster.com/>

<https://www.natashatherobot.com/>

<https://iosdevweekly.com/>

<https://indieiosfocus.com/>

<https://cocoapods.org/>

<https://www.cocoacontrols.com/>

MERCI !!!

