1

Un petit Sondage

- Qui a déjà fait du Java ?
- Qui en fait encore régulièrement ?
- Qui aime en faire ?

2

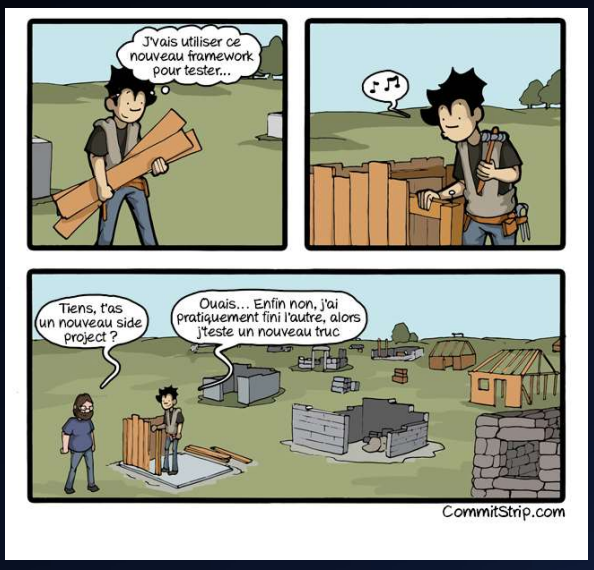
Qui suis-je ?

Arthur Veys @FofoxFRA
Développeur Full-Stack

- 2 ans et demi en start-up
- Présentement développeur backend chez SM360

3

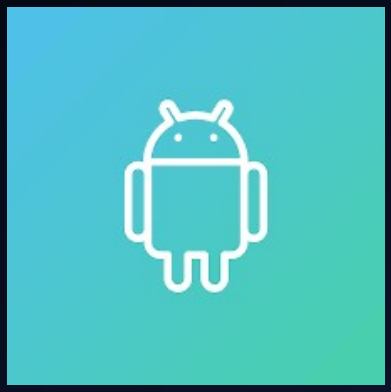
Une passion :



4

Quand je parle de Kotlin, on me dit souvent ….

Hey mais c'est le langage d'Android !



5

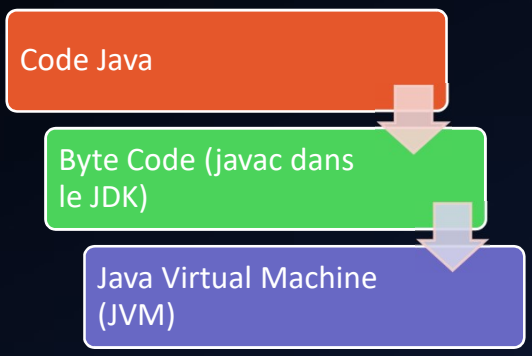
Workshop Kotlin

UN LANGAGE POUR LES REMPLACER TOUS ?

6

Au commencement ….

- Java 1.0 -> 1996
  - La promesse : ***Write Once, Run Anywhere***



- Top 3 des langages les plus utilisés
- Version 13 sortie en septembre 2019

7

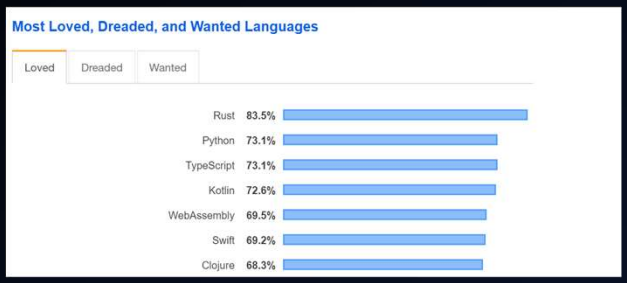
Kotlin

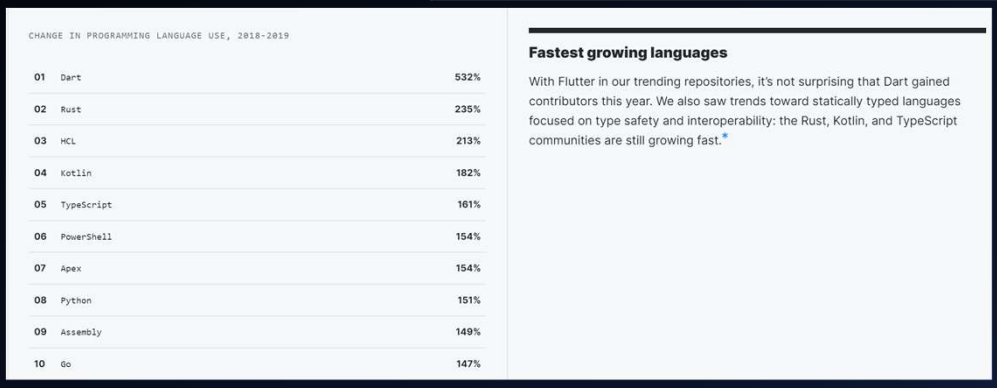
- Projet démarré en 2010 chez Jetbrains (IntelliJ)
  - Le but : faire un langage plus typé et robuste que java, moins verbeux que Java
    - En gros, un java moderne ! (Typescript vs Javascript)
  - Inspiration : Scala, C#, Swift, Javascript
  - Nommé par rapport à une île de St-Peterbourg
  - Stable depuis 2016 (v1.3 fin octobre 2018)
  - Supporté par Google pour Android en 2017 puis annoncé comme langage principal en 2019
  - Supporté officiellement par Spring 5 fin 2017

8

La Hype

Source : StackOverflow Insight 2019
90 000 réponses

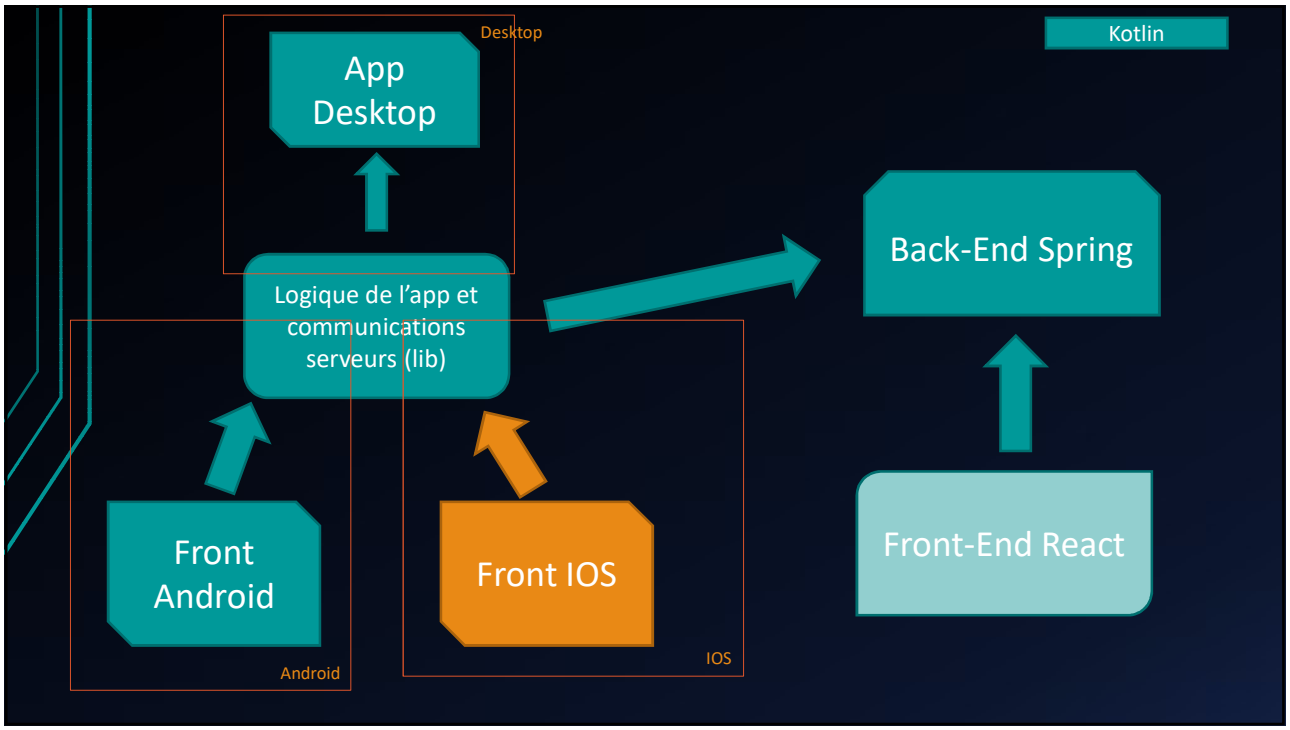




Source : Github Octoverse 2019

La magie

- Kotlin est complétement interopérable avec Java
  - Les deux langages peuvent cohabiter dans le même projet
  - On peut appeler des classe Kotlin en Java et Inversement
- Kotlin est multiplateforme à plus haut niveau que Java grâce à LLVM
  - Transpilation du Kotlin en JS
  - Compilation en code Natif et interopérable C, Swift, cocoaPods (Win32, Linux, Mac, IOS)
  - Compilation en WebAssembly



11

En pratique

12

- ; optionnel en fin de ligne
- Les fonctions sont au même niveau que les classes

```
class SquareGrid(width: Int, height: Int, barriers: List<Barrier>) : Grid(barriers) {

    private val heightRange: IntRange = (0 until height)
    private val widthRange: IntRange = (0 until width)

    private val validMoves = listOf(Pair(1, 0), Pair(-1, 0), Pair(0, 1), Pair(0, -1), Pair(1, 1), Pair(-1, 1), Pair(1, -1), Pair(-1, -1))

    override fun getNeighbours(position: GridPosition): List<GridPosition> = validMoves
        .map { GridPosition(position.first + it.first, position.second + it.second) }
        .filter { inGrid(it) }

    private fun inGrid(it: GridPosition) = (it.first in widthRange) && (it.second in heightRange)
}
fun aStarSearch(start: GridPosition, finish: GridPosition, grid: Grid): Pair<List<GridPosition>, Int> {
    fun generatePath(currentPos: GridPosition, cameFrom: Map<GridPosition, GridPosition>): List<GridPosition> {
        val path = mutableListOf(currentPos)
        var current = currentPos
        while (cameFrom.containsKey(current)) {
            current = cameFrom.getValue(current)
            path.add( index: 0, current)
        }
        return path.toList()
    }
    val openVertices = mutableSetOf(start)
    val closedVertices = mutableSetOf<GridPosition>()
    val costFromStart = mutableMapOf(start to 0)
    val estimatedTotalCost = mutableMapOf(start to grid.heuristicDistance(start, finish))

    val cameFrom = mutableMapOf<GridPosition, GridPosition>()  // Used to generate path by back tracking

    while (openVertices.size > 0) {...}

    throw IllegalArgumentException("No Path from Start $start to Finish $finish")
}

fun main() {

    val barriers = listOf(setOf( Pair(2,4), Pair(2,5), Pair(2,6), Pair(3,6), Pair(4,6), Pair(5,6), Pair(5,5),
        Pair(5,4), Pair(5,3), Pair(5,2), Pair(4,2), Pair(3,2)))

    val (path, cost) = aStarSearch(GridPosition(0,0), GridPosition(7,7), SquareGrid( width: 8, height: 8, barriers))

    println("Cost: $cost  Path: $path")
}
```


Variables & types & fonctions

```
fun main() {
    val number:Int = 42
    var text:String = "Hello"
    text += "World"
    val listName: MutableList<String> = mutableListOf("ceci","est","un","tableau")
    println("Number = $number / Sum : ${sum(number, 6)} / div : ${div(number, 4)} / List: $listName")
}

fun sum(a:Int, b:Int):Int {
    return a + b
}

fun div(a: Int, i: Int) = a/i
```

Augmented assignments

Expression	Translated to
a += b	a.plusAssign(b)
a -= b	a.minusAssign(b)
a *= b	a.timesAssign(b)
a /= b	a.divAssign(b)
a %= b	a.modAssign(b)

Arithmetic operators

Expression	Translated to
a + b	a.plus(b)
a - b	a.minus(b)
a * b	a.times(b)
a / b	a.div(b)
a % b	a.rem(b), a.mod(b) (deprecated)
a..b	a.rangeTo(b)

'In' operator

Expression	Translated to
a in b	b.contains(a)
a !in b	!b.contains(a)

Equality and inequality operators

Expression	Translated to
a == b	a?.equals(b) ?: (b === null)
a != b	!(a?.equals(b) ?: (b === null))

Indexed access operator

Expression	Translated to
a[i]	a.get(i)
a[i, j]	a.get(i, j)
a[i_1, ..., i_n]	a.get(i_1, ..., i_n)
a[i] = b	a.set(i, b)
a[i, j] = b	a.set(i, j, b)
a[i_1, ..., i_n] = b	a.set(i_1, ..., i_n, b)

Operator Overloading

```kotlin
data class Point(val x: Int, val y: Int)

operator fun Point.dec() = Point(-x, -y)

var point = Point(10, 20)

fun main() {
    println(point--)  // prints "Point(x=-10, y=-20)"
}
```

15

List

- Par défaut les Lists sont Read-Only. MutableList pour les rendre mutables
- Accès aux valeurs grâce à []

```kotlin
println(list[1])
println(list["avion"])
```

16

Super Types

- Super type : Any (Object en Java) & Nothing (Void)
- Pour les retours de fonctions, 2 types remplacent Void
  - Unit -> le retour de la fonction est sans intérêt (souvent implicite)

```
fun test(): Unit {
    print("hello")
}
```

```
fun test(){
    print("hello")
}
```

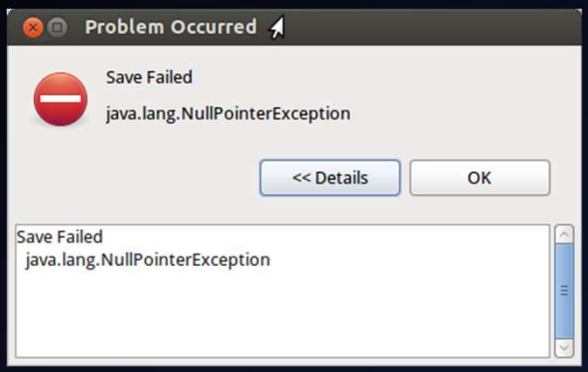
  - Nothing -> La fonction ne retourne jamais (aucune valeur retournée)

```
fun fail(message:String): Nothing {
    throw IllegalArgumentException(message)
}
```

    - TODO est une fonction qui renvoi Nothing

17

Le cauchemar des devs Java ?



18

Nullables Types

- Faire en sorte que les NPE deviennent des erreurs de compilation au lieu de runtime
- Un type spécifique pour le nullable valable pour tout type existant
- Utilisation de l'opérateur ? Et !!

```
val errorText:String = null
val text:String? = "null or not null ? That's the question"

errorText.length
text!!.length
```

- Elvis operator ou Smart Cast

```
val text:String? = " null or not null ? That's the question"
val size:Int? = text?.length
println(text ?: "Null value")
```

19

Structures conditionnelle

```
val x:Any = 2
val y:String = when(x) {
    0 -> "Zero" // Equality check
    in 1..4 -> "Four or less" // Range check
    5, 6, 7 -> "Five to seven" // Multiple values
    is Byte -> "Byte" // Type check
    else -> "Some number"
}
```

```
val a = 4
val b = 6
val max = if (a > b) {
    print("Choose a")
    a
} else {
    print("Choose b")
    b
}
```

20

Structure Itérative

```
for(i in list){
    print(i)
}

for (i in 1..3) {
    println(i)
}

for (i in list.indices) {
    println(list[i])
}

val map = HashMap<Int, String>()
for((key, value) in map){
    println("$key -> $value")
}
```

21

Classes

- https://speakerdeck.com/svtk/3-object-oriented-programming-in-kotlin-kotlin-workshop
- Génériques et reflectivité possible

22

Fonctionnel

- https://speakerdeck.com/svtk/5-functional-programming-kotlin-workshop
- https://speakerdeck.com/svtk/6-the-power-of-inline-kotlin-workshop?

23

Async & Coroutine

- Async / Await pour les requêtes à des services

```
fun loadImage(url:String) = async{…}

//Async function
val image = loadImage(url).await()
setImage(image)
```

- Coroutine : Thread ultra-léger qui peut-être suspendu
  - Async -> thread qui peut être suspendu
  - Await -> Suspend le traitement

24

Sources et tutoriels

- https://kotlinlang.org/docs/reference/ : Doc Officielle
- https://github.com/Kotlin/workshop : Le Github du workshop
- https://play.kotlinlang.org/ : Tuto et compilateur en ligne

25

2 CHOIX POSSIBLES

Live Coding en mode Questions/Réponses

- Un mini projet :
  - Gérer un µS de stations météo
- Entité
  - Stations météo
  - Relevé
- Endpoints
  - CRUD relevé
  - CRUD Stations
  - Quelques agrégations

26

Le projet

https://github.com/Aveys/ktor-meteo