

Développement Android avec le langage Kotlin

 [Macha DA COSTA](#)

 [@MachaDaCosta](#)



Introduction à Kotlin

G I/O  17-19 Mai 2017



 Officialisation du langage Kotlin

Introduction à Kotlin

</> Kotlin de JetBrains



.....



.....



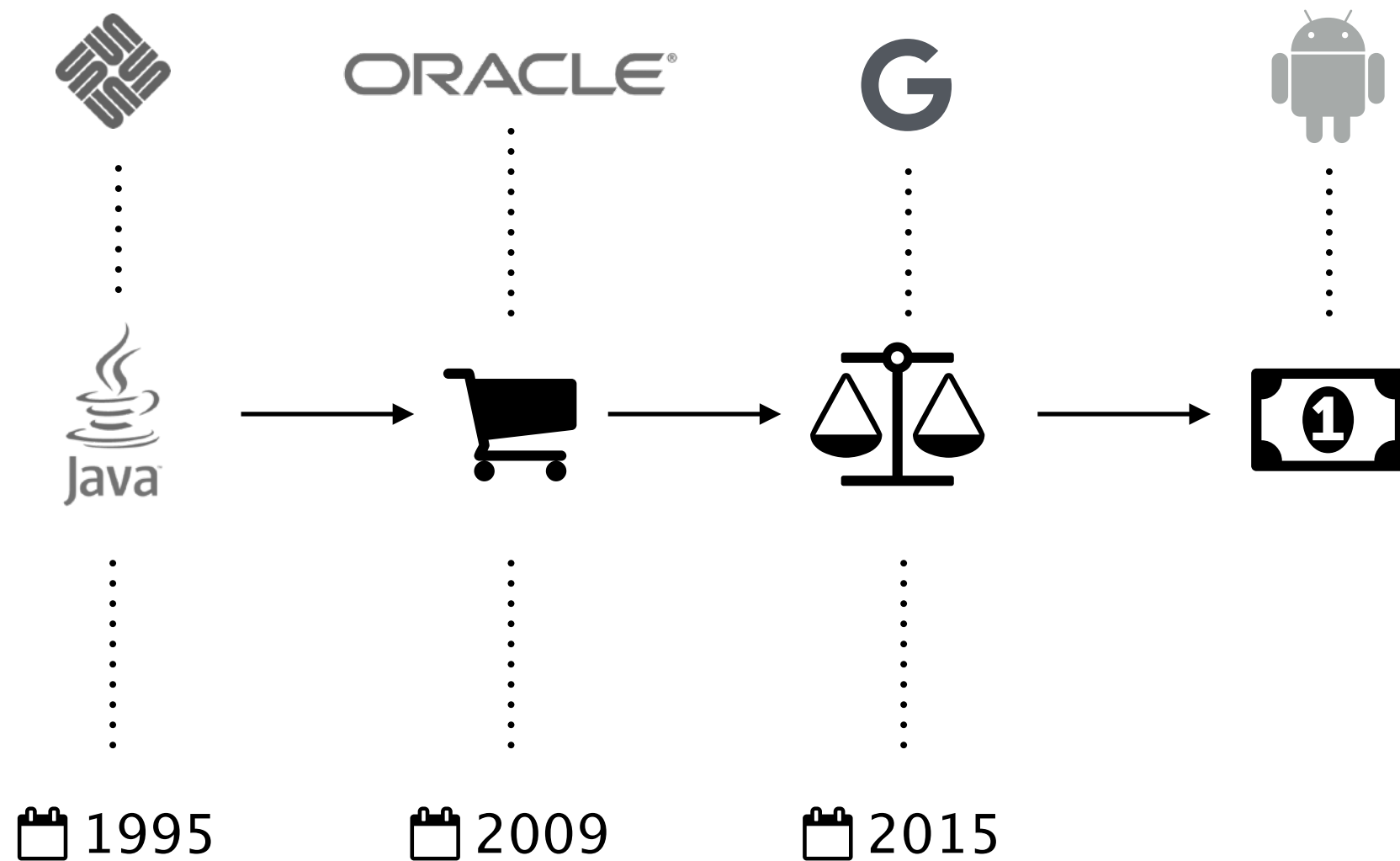
Introduction à Kotlin

G Pourquoi Kotlin ?



Introduction à Kotlin

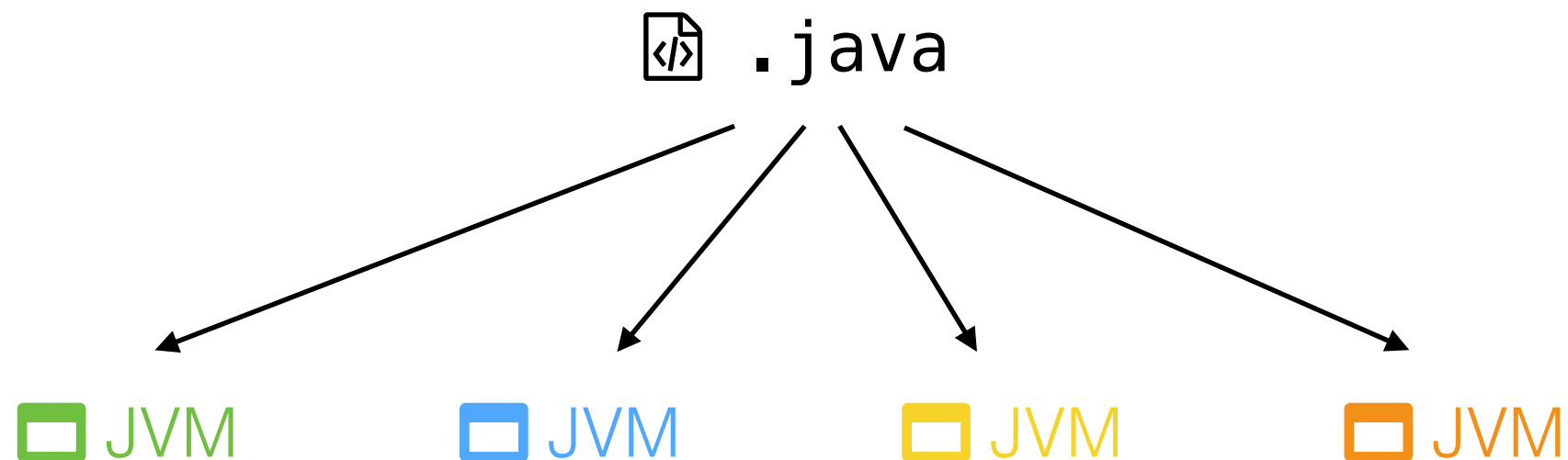
Pourquoi Kotlin ?



💡 L'histoire de Java

Introduction à Kotlin

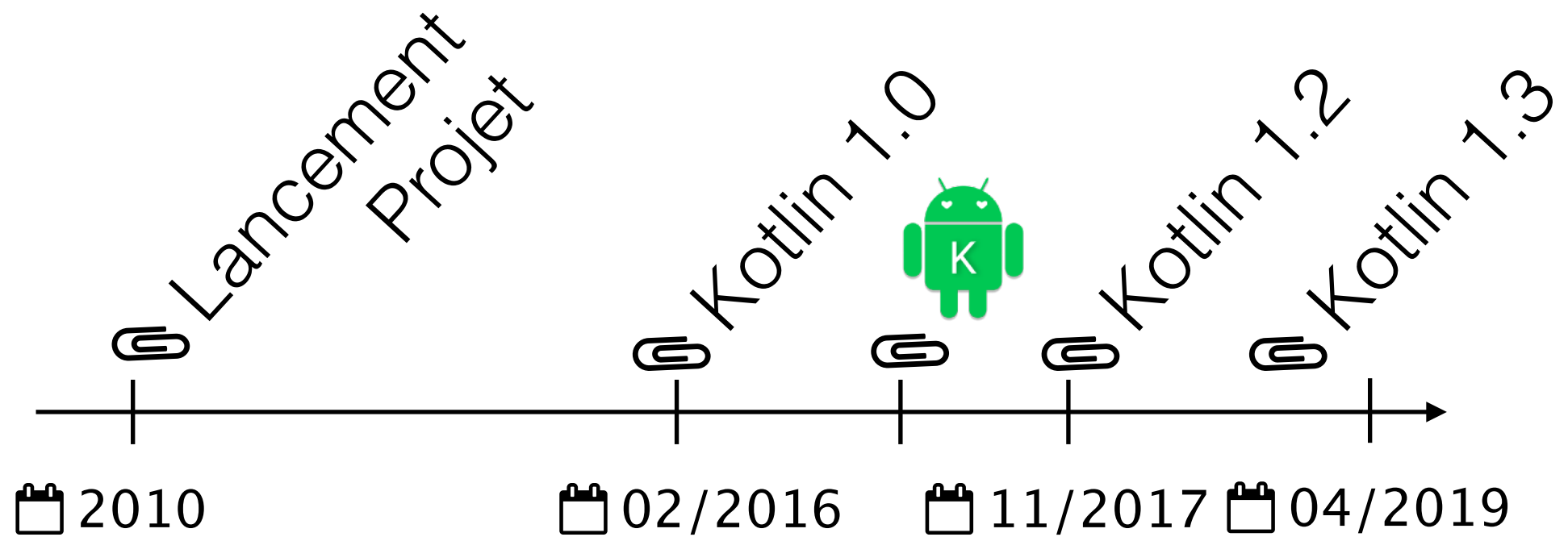
Pourquoi Kotlin ?



Le concept de Java

Introduction à Kotlin

</> Historique



Introduction à Kotlin

 Kotlin \approx Swift

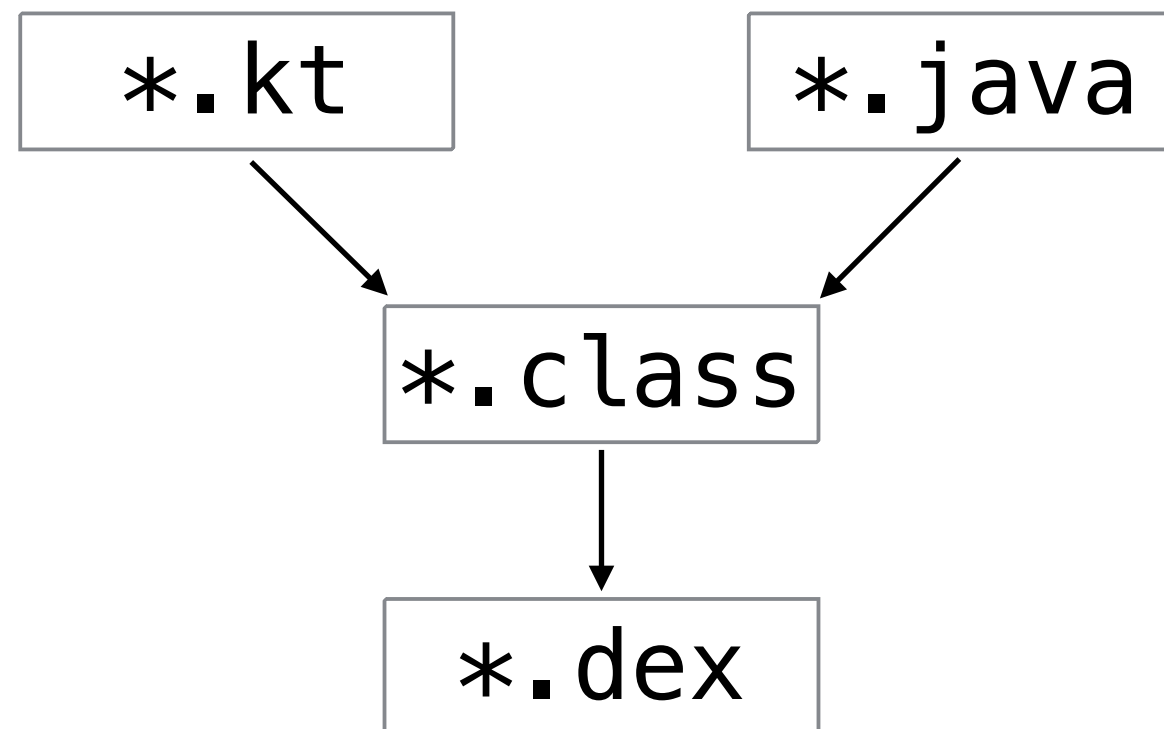
 iOS : Objective-C vers Swift

 WWDC 2014

Introduction à Kotlin

</> Concept de Kotlin

📎 Compilé en bytecode Java



Introduction à Kotlin

♥ Android et Kotlin

Conversion automatique

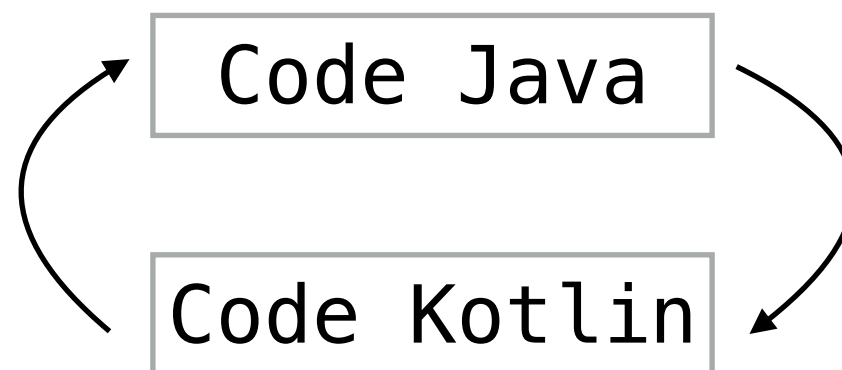
`.java : Ctrl C → .kt : Ctrl V`

`'Code' > 'Convert Java File to Kotlin File'`
`(Command + Option + Shift + K)`

Introduction à Kotlin

♥ Android et Kotlin

📎 Cohabitation Java & Kotlin



Introduction à Kotlin

Kotlin Multiplateforme

Interopérabilité

Plateformes supportées

iOS (arm32, arm64, simulator x86_64)

macOS (x86_64), watchOS (arm32, arm64, x86), tvOS (arm64, x86_64)

Android (arm32, arm64, x86, x86_64)

Windows (mingw x86_64, x86)

Linux (x86_64, arm32, MIPS, MIPS little endian, Raspberry Pi)

WebAssembly (wasm32)

Code Partagé

Android

iOS

JavaScript, CSS

Introduction à Kotlin

</> Kotlin en bref

📎 Langage clair et concis

`;` → 😊

~~getter/setter~~

`Log.i(TAG, "my width is $mWidth")`

Boilerplate → *Lambdas*

📎 Affranchissement du `NullPointerException`

📎 Programmation orientée objet et fonctionnelle



Kotlin

Macha
DA COSTA

Site Web

*macha@CHILL***CODING**.com

Email

Développement Android avec Kotlin

A. Classe







B. Variables, Opérateurs, Conditions

C. Programmation fonctionnelle

Kotlin pour l'Interface Utilisateur

A.

Classe

-  Déclaration
-  Propriété
-  Objet compagnon
-  Fonction
-  Qualificatifs
-  Héritage

A.

Classe

Déclaration :   MagicCircle

```
public class MagicCircle {  
    private final int cx;  
    private final int cy;  
  
    public MagicCircle(int cx, int cy) {  
        this.cx = cx;  
        this.cy = cy;  
    }  
  
    public int getCx() {  
        return cx;  
    }  
  
    public int getCy() {  
        return cy;  
    }  
}
```

 Convert Java to Kotlin



</> Création d'une classe représentant un cercle

A.

Classe

Déclaration

```
data class MagicCircle(val cx: Int, val cy: Int)
```



i gérer des données

💡 equals

💡 hashCode

💡 toString

A.

Classe

Déclaration : constructeur primaire

```
class MagicCircle(val cx: Int, val cy: Int)
```

i dans la déclaration
de la classe



A.

Classe

Déclaration : constructeur secondaire

```
class MagicCircle {  
    var cx = 0  
    var cy = 0  
  
    constructor(cx:Int, cy:Int) {  
        this.cx = cx  
        this.cy = cy  
    }  
}
```

i explicite
dans la classe

A.

Classe

Propriété

```
public class MagicCircle {  
    private final int cx;  
    private final int cy;  
  
    public MagicCircle(int cx, int cy) {  
        this.cx = cx;  
        this.cy = cy;  
    }  
  
    public int getCx() {  
        return cx;  
    }  
  
    public int getCy() {  
        return cy;  
    }  
}
```

magic.getCx()



```
class MagicCircle(  
    val cx: Int,  
    val cy: Int  
)
```

magic.cx



A.

Classe

Exemple

```
var magic = MagicCircle(0, 0)
magic.cx = 26
```

i *Ø new*

i *getters/setters implicite*

A.

Classe

Objet compagnon

```
companion object {  
    val FRAGMENT_ABOUT = 1  
    val base64EncodedPublicKey = "MY_BIDUL_KEY"  
}
```

i constantes de classe

A.

Classe

Fonction

```
data class MagicCircle(var cx: Int, var cy: Int) {  
    fun move() = cx += 1  
}
```

i expression simple

A.

Classe

Fonction : paramètre

```
data class MagicCircle(val maxX: Int, val maxY: Int) {  
  
    var cx = 0  
    var cy = 0  
  
    fun move(number: Int) {  
        cx += number  
        cy += number  
    }  
}
```

A.

Classe

Fonction : type de retour

```
data class MagicCircle(val maxX: Int, val maxY: Int) {  
  
    ...  
  
    fun move(number: Int): Boolean {  
        cx += number  
        cy += number  
        return true  
    }  
}
```

A.

Classe

Fonction

```
data class MagicCircle(val maxX: Int, val maxY: Int) {  
  
    ...  
  
    fun move(number: Int = 5) {  
        cx += number  
        cy += number  
    }  
}
```

i argument par défaut

A.

Classe

Fonction

```
data class MagicCircle(val maxX: Int, val maxY: Int) {  
  
    ...  
  
    fun move(dx: Int = 5, dy: Int = 5) {  
        cx += dx  
        cy += dy  
    }  
}
```

i plusieurs arguments

A.

Classe

Qualificatifs

data

enum

inner

sealed

❗ final, open,
abstract par défaut

❗ public, private,
internal par défaut

A.

Classe

Héritage

```
class MainActivity : AppCompatActivity()
```

i extends -> :

B. Variables, Opérateurs, Conditions

- 📎 Déclaration
- 📎 Nullité sécurisée
- 📎 Différents types
- 📎 Opérateurs
- 📎 Conditions
- 📎 Boucle

B. Variables, Opérateurs, Conditions

Déclaration de variable Mutable ou Immuable

```
var mVariable = 10  
val mConstant = 10 // ♥ Préférence pour l'Immuable
```

i var comme variable

B. Variables, Opérateurs, Conditions

Déclaration du type optionnelle

```
var mX = 10 // Conversion de type implicite  
var mHello: String = "Hello you"
```

B. Variables, Opérateurs, Conditions

Déclaration de type `Unit`

```
val mObject: Unit = Unit // Objet vide
```

i \approx void

B. Variables, Opérateurs, Conditions

Nullité sécurisée

 Rien n'est *null*

✗ `var s1 = null` // Ne compile pas !

 C'est quand même possible

✓ `var s2:String? = null` // «?» signifie peut être nul

 Vérification avant compilation

✗ `s2.length`

 Il est peut être *null*

✓ `s2?.length` // Ça revient à `if(s2 != null) {s2.length}`

 Il est déjà initialisé

✓ `s2!!.length` // Si c'est pas le cas -> Exception `NullPointerException`

B. Variables, Opérateurs, Conditions

Différents types

Double (64)

Float (32)

Long (64)

Int (32)

Short (16)

Byte (8)

 Les nombres

B. Variables, Opérateurs, Conditions

Différents types

```
val mX = 20 // Conversion implicite en Int  
val mY: Float = mX.toFloat() // Conversion explicite en Float  
val z = 10F // Conversion implicite en Float
```

i conversion de nombres

B. Variables, Opérateurs, Conditions

Différents types

String
Boolean
Char
Array

 Les classiques

B. Variables, Opérateurs, Conditions

Différents types

```
val s = "abc"  
val str = "$s length is ${s.length}" // evaluates to "abc length is 3"
```

i interpellation de *String*

B. Variables, Opérateurs, Conditions

Différents types

List

MutableList

ArrayList

IntArray

 Les collections

B. Variables, Opérateurs, Conditions

Différents types : Tableau

```
var mArray = Array(6, { i: Int -> MagicCircle(i, i) })
```

B. Variables, Opérateurs, Conditions

Différents types : Tableau

```
var mArray = arrayOf("Jake", "Jill", "Ashley", "Bill")
```

B. Variables, Opérateurs, Conditions

Différents types : Liste

```
var mutableArray: MutableList<MagicCircle> = ArrayList()
```

B. Variables, Opérateurs, Conditions

Différents types : correspondances avec Java

Double

double

Float

float

Int

int

Boolean

boolean

Double?

java.lang.Double

Float?

java.lang.Float

Int?

java.lang.Integer

String?

String

Boolean?

java.lang.Boolean

IntArray

int[]

Array<Int>

Integer[]



B. Variables, Opérateurs, Conditions

Opérateurs

 Unaire

 Binaire

 Tableau

 Égalité

B. Variables, Opérateurs, Conditions

Opérateurs

Unary prefix operators

Expression	Translated to
<code>+a</code>	<code>a.unaryPlus()</code>
<code>-a</code>	<code>a.unaryMinus()</code>
<code>!a</code>	<code>a.not()</code>

 Unaire

B. Variables, Opérateurs, Conditions

Opérateurs

Increments and decrements

Expression	Translated to
<code>a++</code>	<code>a.inc()</code> + see below
<code>a--</code>	<code>a.dec()</code> + see below

 Unaire

B. Variables, Opérateurs, Conditions

Opérateurs

Arithmetic operators

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

 Binaire

B. Variables, Opérateurs, Conditions

Opérateurs

```
if (a >= 0 && a <= 10)
```

≈

```
if(a in 0..10 )
```

 Binaire

 inclusif !

B. Variables, Opérateurs, Conditions

Opérateurs

'In' operator

Expression	Translated to
<code>a in b</code>	<code>b.contains(a)</code>
<code>a !in b</code>	<code>!b.contains(a)</code>

 Binaire

B. Variables, Opérateurs, Conditions

Opérateurs

Augmented assignments

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

 Binaire

B. Variables, Opérateurs, Conditions

Opérateurs

Indexed access operator

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

 Tableau

B. Variables, Opérateurs, Conditions

Opérateurs

Equality and inequality operators

Expression	Translated to
<code>a == b</code>	<code>a?.equals(b) ?: (b === null)</code>
<code>a != b</code>	<code>!(a?.equals(b) ?: (b === null))</code>

 Égalité

B. Variables, Opérateurs, Conditions

Opérateurs

Comparison operators

Expression	Translated to
<code>a > b</code>	<code>a.compareTo(b) > 0</code>
<code>a < b</code>	<code>a.compareTo(b) < 0</code>
<code>a >= b</code>	<code>a.compareTo(b) >= 0</code>
<code>a <= b</code>	<code>a.compareTo(b) <= 0</code>

 Comparaison

B. Variables, Opérateurs, Conditions

Opérateurs

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    var tv = findViewById(R.id.mainTextView) as TextView  
}
```

 Conversion

B. Variables, Opérateurs, Conditions

Conditions

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    when (item.itemId) {  
        R.id.action_settings -> {  
            showAlert()  
            return true  
        }  
        R.id.action_info -> {  
            showInfo()  
            return true  
        }  
        else -> return super.onOptionsItemSelected(item)  
    }  
}
```


B. Variables, Opérateurs, Conditionss

Conditions

```
when (view) {  
  is TextView -> toast(view.text)  
  is RecyclerView -> toast("Item count : ${view.adapter.itemCount}")  
  is SearchView -> toast("Current query: ${view.query}")  
  else -> toast("View type not supported")  
}
```

i exclusif !

B. Variables, Opérateurs, Conditionss

Conditions : *range*

```
val res = when {  
  x in 1..10 -> "cheap"  
  s.contains("hello") -> "it's a welcome!"  
  v is ViewGroup -> "child count: ${v.getChildCount()}"  
  else -> ""  
}
```

i variable dans un rang

B. Variables, Opérateurs, Conditions

Conditions : le classique

```
if (dx < 0)
    cx = 0F
else
    cx = maxX.toFloat()
```

B. Variables, Opérateurs, Conditions

Conditions : le ternaire

`s2?.length ? : 0`

B. Variables, Opérateurs, Conditions

Boucle

```
val names = arrayOf("Jake", "Jill", "Ashley", "Bill")
for (name in names) {
    toast(name)
}
```

Kotlin pour l'Interface Utilisateur

Configuration de Vue

Gestion du clic

Élément Graphique Natif avec la bibliothèque Anko

Sources

Kotlin pour Android

- [TRY Kotlin](#)
- [Kotlin Slides and Questions](#)
- <https://antonioleiva.com/free-kotlin-android-course/>
- <https://www.chillcoding.com/blog/2017/10/03/ajouter-extensions-kotlin/>
- <https://www.chillcoding.com/blog/2017/09/28/configurer-kotlin-projet-android/>

Bibliothèque Anko

- <https://github.com/kotlin/anko>
- <https://www.kotlinddevelopment.com/why-should-use-anko/>
- <https://antonioleiva.com/dialogs-android-anko-kotlin/>
- [AK 4 : Utiliser-anko-kotlin-android](#)

Fonction d'extension

- [Les fonctions d'extension de Kotlin](#)
- <http://tutos-android-france.com/introduction-a-kotlin/>
- [Vidéo de Jake Wharton sur Kotlin \(DEC 2015\)](#)
- [Vidéo de Huyen Tue Dao & Christina Lee sur The Road to Kotlin town \(KotlinConf 2017\)](#)

Sources

Pourquoi Kotlin ?

- [Apple : la fin d'Objective-C au profit de SWIFT ?](#)
- [Swift is like Kotlin](#)
- [Langage Java](#)
- [API Java : Google a enfreint les brevets d'Oracle, selon la Cour Suprême](#)
- [Antoniroleiva: 12 reasons to strat Kotlin for Android](#)
- [ChillCoding : Introduction à Kotlin](#)

Kotlin en bref

- [Kotlin: pourquoi ce nouveau langage est une bonne nouvelle](#)
- [Introduction to Kotlin Google I/O '17](#)
- [Kotlin it's the little things](#)
- [Android Development with Kotlin](#)

Android et Kotlin

- [Android Studio 3.0 Canary](#)
- [Kotlin - Official Site](#)
- [developer.android: Get Started with Kotlin on Android](#)

Sources

Type Kotlin

- <https://code.tutsplus.com/tutorials/kotlin-from-scratch-variables-basic-types-arrays-type-inference-and-comments--cms-29328>
- <https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-array/index.html>
- <http://kotlinlang.org/docs/reference/basic-types.html#arrays>

try.kotlinlang.org

The screenshot shows a web browser window with the URL `try.kotlinlang.org`. The page features the Kotlin logo and navigation links for [LEARN](#), [COMMUNITY](#), and [TRY ONLINE](#). Below the navigation bar, there are social media icons for Facebook, Google+, GitHub, and JetBrains, along with utility buttons for [Shortcuts](#), [Convert from Java](#), and [Fullscreen](#).

The main content area is titled "Kotlin Koans" and includes a breadcrumb trail: [Kotlin Koans](#) > [Introduction](#) > [Hello, world!](#) > [Task.kt](#). A sidebar on the left lists various koan categories, with "Introduction" expanded to show "Hello, world!" and "Task.kt" selected. The "Task.kt" file is highlighted in blue.

The main task area is titled "Simple Functions" and contains the following text:

Take a look at [function syntax](#) and make the function `start` return the string `"OK"`.

In the tasks the function `TODO()` is used that throws an exception. Your job during the koans will be to replace this function invocation with a meaningful code according to the problem.

Below the text are three buttons: [Check](#), [Revert](#), and [Show answer](#).

The code editor shows the following Kotlin code:

```
1 fun start() = "OK"
```

CustomView

- Data class MagicCircle
- CustomView avec cercles
- Tableau de cercles
- Fonction d'extension Random

IF YOU THINK
YOU ARE TOO
SMALL
TO MAKE A
DIFFERENCE
TRY SLEEPING
WITH A MOSQUITO.

_ African Proverb