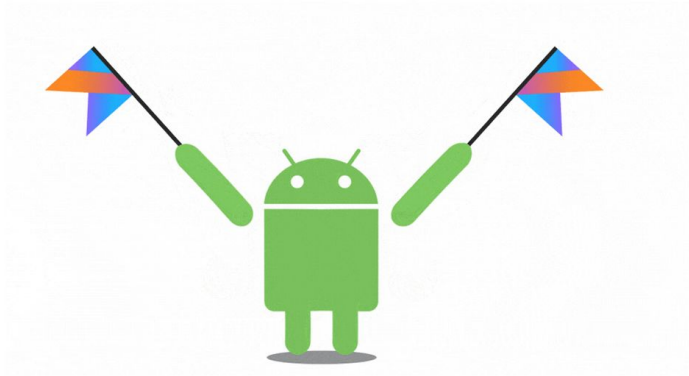
Brown Bag Lunch Kotlin

Yann Mougenel

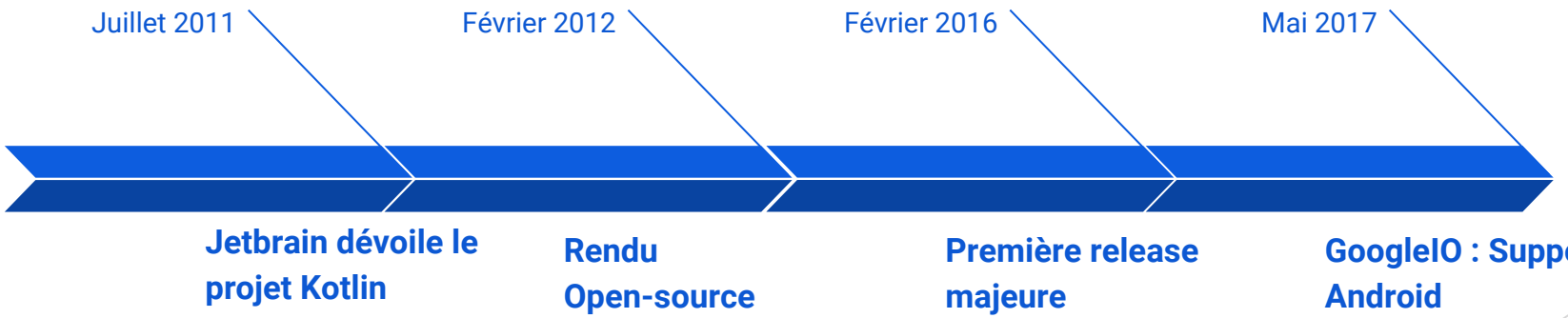
Plan

1. Présentation
   - Kotlin brièvement
   - Quelques chiffres
   - Environnement Java/JVM
2. Syntaxe et Fonctionnalités
   - Moins syntaxique
   - Plus optimisé
   - Fonctionnalités
3. Migration
   - Contexte du projet
   - Déroulement
   - Résultats
4. Conclusion



Présentation
Kotlin brièvement

- Langage de programmation orienté objet, compilé en bytecode
- Application (Java) Back + **Android** + (JavaScript)



v1.0 (licence Apache 2)

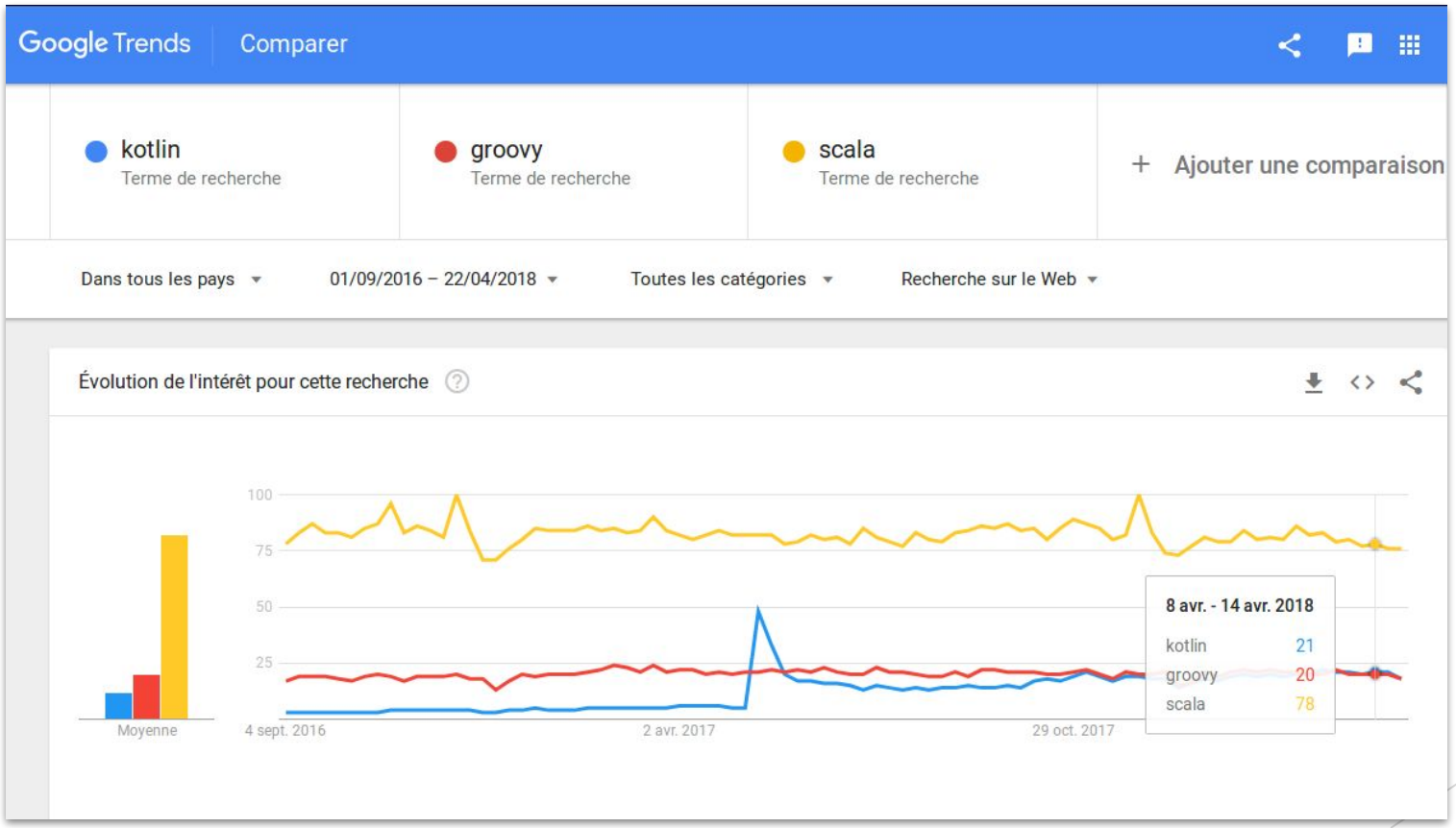
Présentation
Quelques chiffres

Stack Overflow

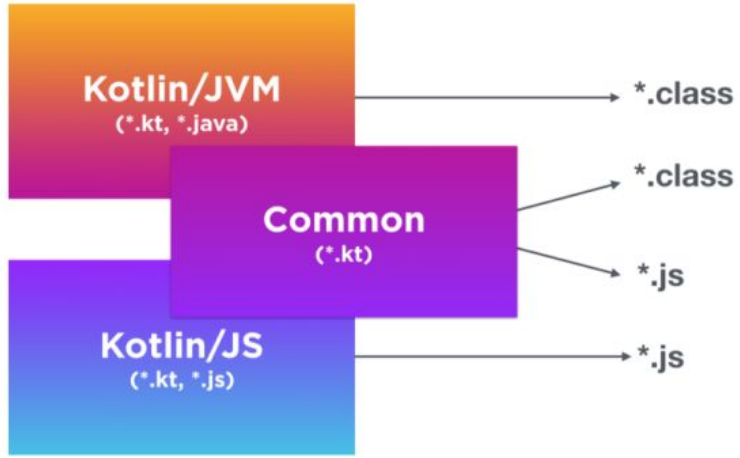
10 000 Topics
2 langage le plus apprécié en 2018

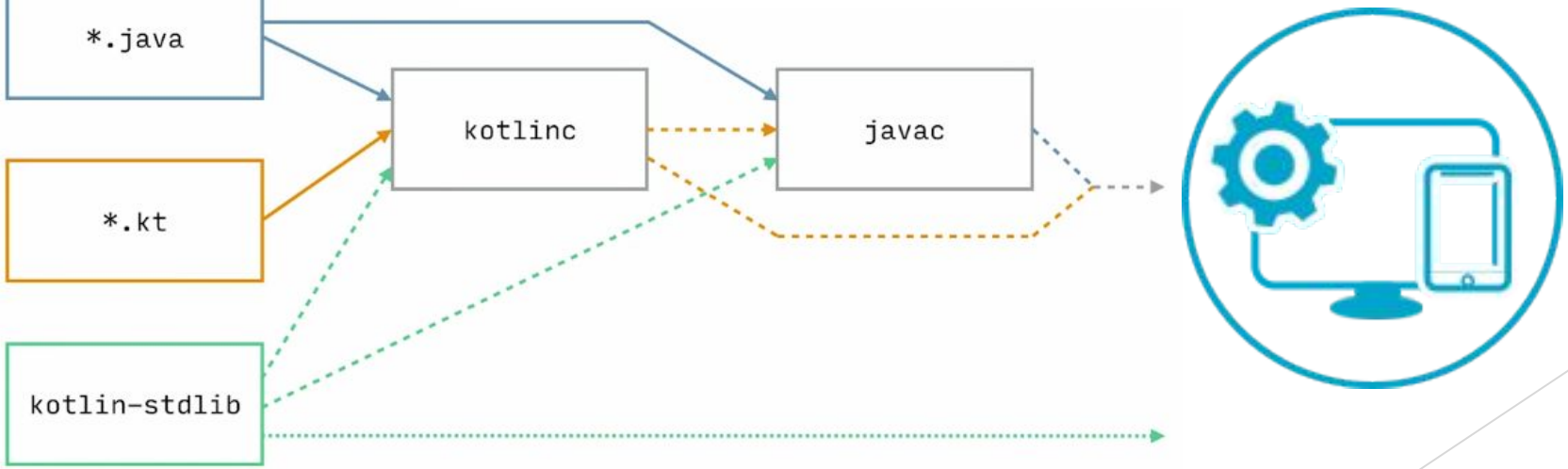
Présentation
Quelques chiffres

Google : 17% des nouveaux projets sous Android Studio 3.0



Présentation
Environnement JVM





Syntaxe et Fonctionnalités

Une syntaxe plus légère

En Java

```
                                2                    2
List<Video> listVideos = new ArrayList<>();
for (Video video : listVideos) {
   System.out.println("Found" + video.getId());
}
```

En Kotlin

```
 1
val listVideos = ArrayList<Video>()
for (video in listVideos) {
   print("Found ${video.id}")
}
     3                4
```

1. val/var : Permet de déclarer des variables
2. Pas de “new” ou de point-virgule
3. De nouvelles fonctions
4. String multiline `var s = “Time is ${LocalDate.now()}”`

   Appel implicite aux accesseurs (getter/setter)

… Et surtout

5. Une inférence de type !

Syntaxe et Fonctionnalités

Une syntaxe plus légère

L'inférence de type pour les variables :

```
var agents : List<Agent> = agentDao.findAll()
for (agent : Agent in agents) {
    …
}
```

=

```
var agents = agentDao.findAll()
for (agent in agents) {
    …
}
```

-> Inclus depuis Java 10

L'inférence de type pour les fonctions :

```
fun double(x : Int) : Int {
   return 2 * x;
}
```

=

```
fun double(x : Int) = 2 * x;
```

Syntaxe et Fonctionnalités

Une syntaxe plus complète

TypeAlias

```
typealias Library = List<Book>

var library : Library = …
```

```
typealias Credentials =
Pair<UserName, PasswordHash>
```

Surcharge des primitifs

```
fun String.removeFirstLastChar(): String =
this.substring(1, this.length - 1)
```

```
val myString= "Hello Everyone"

// affiche “ello Everyon”

myString.removeFirstLastChar();
```

Déconstruction

```
for ((key, value) in map) {
   // do something with   the
key and the value
}
```

Également possible pour les data class

Fonctionnalités
Les DATA class

En java, l'utilisation de modèles (POJO) :

- Attributs privés -> getter/setter
- Constructeurs ou builders
- … C'est tout ?
- Hashcode, equals, clone

→ ~50 lignes de code… assommant

Avec Kotlin, les DATA class :

- Tout est autogénéré

```
data class Utilisateur(@Id var id: String = "", var nom: String, var prenom: String) {
  constructor() : this("", "", "")
}
```

→ ~3 lignes de code \o/

Fonctionnalités
La chasse aux NPE !

Problème récurrent en java : La null Pointeur exception

- En dev -> Oups…
- En recette -> Ah merde…
- En prod -> Oh my god…

Les variables par défaut sur Kotlin ne peuvent pas être nulles :

```
var user : User = null // Ne compile pas
```

- Si ma valeur peut être nulle, je le déclare explicitement :

```
var user : User? = null
Print( user.sister?.name )
```

- A l'utilisation

Fonctionnalités

La chasse aux NPE !

Service

```
user : User = userDao.find(id)
~~if (user != null)~~
```

DAO

findById(id : Int) : User

ORM

SilverBullet?

Autres syntaxes et fonctionnalités…

Egalité entre String (valeur ou référence)

En java : par référence

```
new String("IsEqual") == new String("IsEqual") // False
```

En Kotlin : par valeur

```
new String("IsEqual") == new String("IsEqual") // True
```

Spring 5 : bonus

Syntaxe plus complète si Kotlin est détecté

Valeur par défaut pour les arguments de méthodes

Collections par défaut immutables

Des instructions plus parlantes

```
var age = 15
if (age in 1..10) {...}

for (i in 1..4 step 2) {...}
```

switch sur des types

```
when (x) {
  is Int -> print(x + 1)
  is String -> print(x.length + 1)
  is IntArray -> print(x.sum())
}
```

Smart cast

```
if (x is String) {
  print(x.length) // x is automatically cast to String
}
```

Nom implicit pour une lambda avec 1 paramètre

.filter{ it > 0 }

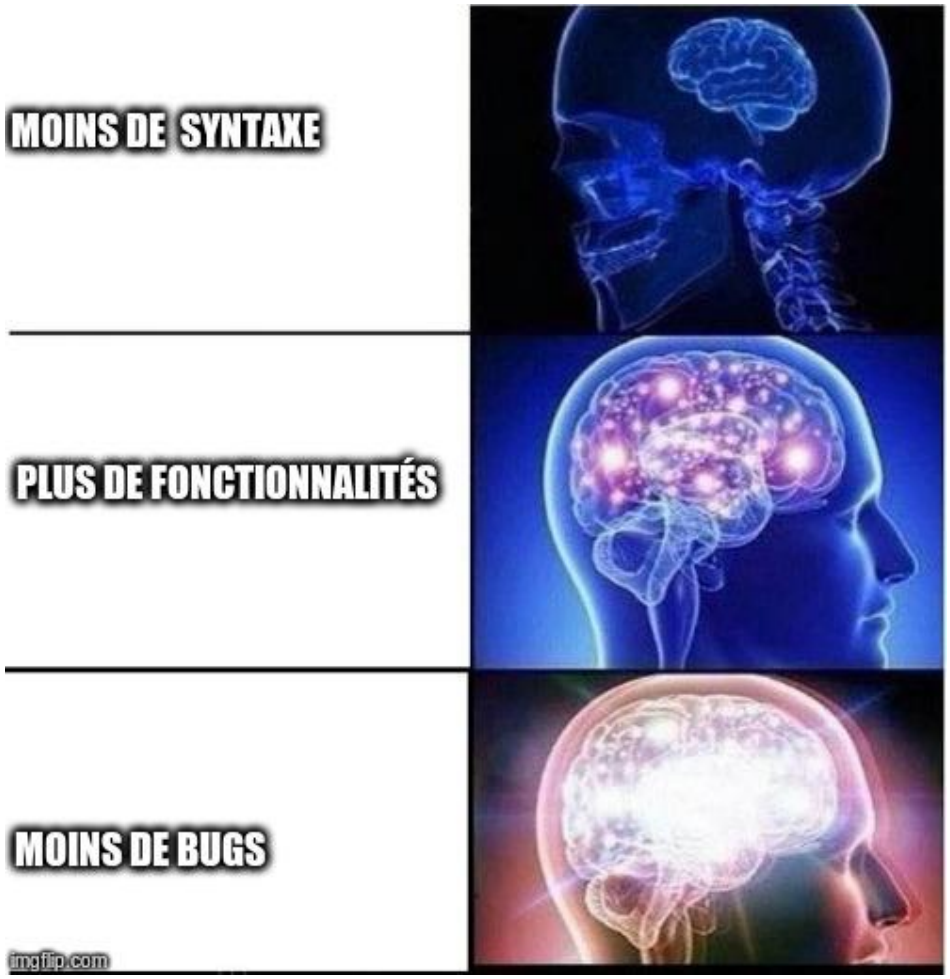
String paramétrable

```
fun greet(name: String) : String {
  return "hello $name"
}
```

Pair : l'arrivée des tuples

```
val (a, b) = Pair(1, "x")
println(a) // 1
println(b) // x
```

Autres syntaxes et fonctionnalités…



Autres syntaxes et fonctionnalités…

Incompatible avec Play Framework

Syntaxe ambiguë
Sysout/print ?
Point virgule à la fin ?

Une syntaxe entièrement objet…plus de static

Migration projet
Pourquoi Kotlin?

Tout le monde en parle

Ils en disent du bien

Migration projet
Quoi ?

ChannelData

Récupération de donnée Youtube

nombre d'abonnés, nombre de vues, like/dislikes…

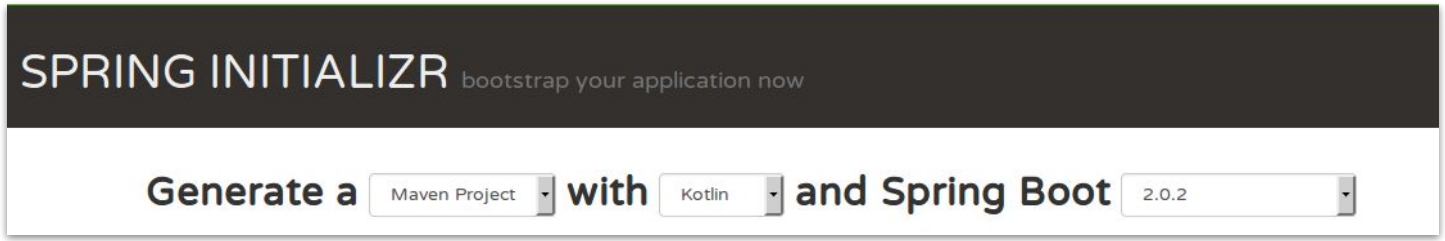
Environnement : SpringBoot 2, Angular, Kotlin

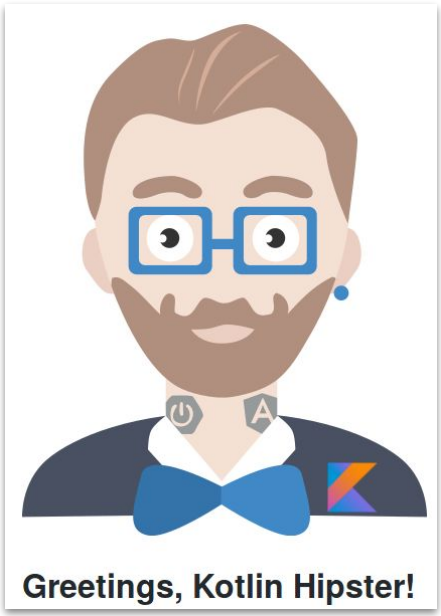
En bref :

- 750 lignes de codes
- 1 Dev
- architecture classique (4 models - 4 daos - 4 contolleurs)

Migration projet
Comment ?

Création d'un projet :





Migration projet
Comment ?

Conversion via intelliJ

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        set
    }
}
```

Enter action or option name:

convert java file to kotlin

Convert Java File to Kotlin File ⌥⇧⌘K Code

Press ^↑ or ^↓ to navigate through the history

+ Ajout d'une pré-compilation maven

Migration projet
Et la… c’est le drame

Problèmes de compilation :

- NPE a traitées
- Static à contourner
- Syntaxe à revoir

Refactor du code :

- DataClass
- Jeu avec la syntaxe

Migration projet
Résultat

Temps passé ~3h
Jurons lancés ~15
Commit faits 8

	Avant	**Après**
Temps de compilation	3 - 4 secondes	7 secondes (100% !)
Nombre de ligne de code	750	690 (470 Kotlin & 220 Java)
NPE potentiels	20	???

Conclusion personnelle

Temps de compilation impacté
Syntaxe nouvelle et ambiguë
Mélange avec java
Paradigme entièrement objet
Partie JS insipide

(Beaucoup) moins verbeux
Des fonctionnalités agréables
Sus aux NPE !
Intègre l'écosystème Java
Courbe d'apprentissage raisonnable

Conclusion générale

Un langage voué à rester (support Android)

Par/Pour les développeurs

Migration partiel recommandée

Références

Information:

- 17% des projets sous android studio
- Documentation Kotlin

Pros:

- Why Kotlin is my next programming language
- Why you should totally switch to Kotlin
- Kotlin for grumpy Java developers
- Devoxx Kotlin for java programmers

Cons:

- The case against kotlin
- Back to java

Others:

- Java 10 Inference

Merci à tous

Questions ?