Kotlin, The Pragmatic Language For Android

Mike Gouline

Android Developer

gouline.net • @mgouline • +MikeGouline

Agenda

- ➔ Background
- ➔ What is Kotlin?
- ➔ Perfect for Android
- ➔ Performance and cost
- ➔ Case study
- ➔ Migration guide
- ➔ Community adoption
- ➔ Current issues
- ➔ Future releases
- ➔ Summary

Background

Background

- Apple got a nice(r) new language
- Android stuck with Java
- Not fair!

Problems with Java

- Missing modern features
  - Lambdas, properties, higher-order functions
- Null safety
  - NullPointerException
- Boilerplate code
- Features specific to JDK (and Android API)

What is Kotlin?

What is Kotlin?

- Named after island in St. Petersburg
- Programming language
  - Based on the JVM
  - Compact and modern (“better Java”)
  - Open source
- Created by JetBrains
  - Built into Android Studio and IntelliJ IDEA
  - Used by JetBrains internally

History

- Project Kotlin unveiled in July 2011
- Kotlin 1.0 released in February 2016
- “Language of the Month” - Dr. Dobb’s Journal (01/2012)

Syntax

- Types follow variable/function names
- Functions start with fun keyword
- Default constructor in class signature
- Semicolons not required

```
class Foo(name: String) : Bar(name) {
   override fun makeStuff(): Stuff {
       return Stuff()
   }
}
```

Null safety

KOTLIN

```kotlin
var str1: String? = null
str1?.trim() // doesn't run

str1 = "Not null anymore"
str1?.trim() // does runs

str1!!.trim() // runs anyway

val str2: String = "I am not null"
str2.trim() // no need for "?."
```

JAVA

```java
String str1 = null;
str1.trim(); // runs and crashes

str1 = "Not null anymore";
str1.trim(); // runs

String str2 = "I am not null";
str2.trim(); // runs
```

Lambdas

KOTLIN

```kotlin
fun evens(nums: List<Int>) = nums.filter { it % 2 == 0 }
```

JAVA

```java
public List<Integer> evens(List<Integer> nums) {
   List<Integer> numsCopy = new ArrayList<>(nums);
   Iterator<Integer> numsItr = numsCopy.listIterator();
   while (numsItr.hasNext()) {
       Integer num = numsItr.next();
       if (num % 2 != 0) numsItr.remove();
   }
   return numsCopy;
}
```

Data classes

KOTLIN

```kotlin
data class Island(var name: String)
```

JAVA

```java
public static class Island {
   private String mName;

   public Island(String name) { mName = name; }
   public String getName() { return mName; }
   public void setName(String name) { mName = name; }

   @Override public boolean equals(Object o) {
       if (this == o) return true;
       if (o == null || getClass() != o.getClass()) return false;
       Island island = (Island) o;
       return mName != null ? mName.equals(island.mName) : island.mName == null;
   }

   @Override public int hashCode() { return mName != null ? mName.hashCode() : 0; }
}
```

Properties in Java code

```
// Java code
public class Circle {
    private float mRadius;
    public float getRadius() { return mRadius; }
    public void setRadius(float radius) { mRadius = radius; }
}

// Kotlin code
val circle = Circle()
circle.radius = 1.5f // => circle.setRadius(1.5f)
println(circle.radius) // => circle.getRadius()
```

Sealed classes (algebraic data types)

```
// Arithmetic expression
sealed class Expr {
    class Const(val number: Double) : Expr()
    class Sum(val e1: Expr, val e2: Expr) : Expr()
    object NotANumber : Expr()
}

fun eval(expr: Expr): Double = when (expr) {
    is Expr.Const -> expr.number
    is Expr.Sum -> eval(expr.e1) + eval(expr.e2)
    Expr.NotANumber -> Double.NaN
}
```

Named/optional arguments

```
// Argument "stroke" is optional
fun circle(x: Int, y: Int, rad: Int, stroke: Int = 1) {
    ...
}

// Argument "rad" is named and "stroke" defaults to 1
circle(0, 0, rad = 5)
```

Extension functions

```
// Extension to String
fun String.encodeSpaces(): String {
    return this.replace(" ", "_")
}

println("one two three".encodeSpaces()) // output: one_two_three
```

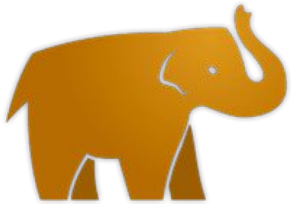
Perfect for Android

Perfect for Android

- Android stuck with Java 6 or 7 (depending on API)
- Complete interop with Java
- Compact runtime
- Do more with less code

Why not others?

- Scala
  - Huge runtime
  - Lots of garbage collection
- Groovy
  - Large runtime
  - Average tooling support
- Ceylon
  - Not much support for Android



Android extensions

- View binding (like Butter Knife)
- No instance variables required
- How?
  - Import synthetic layout
    - `import kotlinx.android.synthetic.main.<layout>.*`
  - Use view by ID
    - E.g. `txt_status.text = "OK"`
  - Under the hood: synthetic calls replaced by functions

Android extensions

```
import kotlinx.android.synthetic.main.activity_main.*

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    btn_go.setText(R.string.go)
    btn_go.setOnClickListener { v ->
        txt_status.text = "Done"
    }
}
```

Nullability

- Remember nullable types, e.g. `String` vs `String?`
- Compatible with `@NonNull` and `@Nullable` annotations
  - `@NonNull` → `String`
  - `@Nullable` → `String?`
- Works with `@Inject` annotation
  - `@Inject lateinit val foo: Foo`
  - Non-nullable, even though not instantiated

Annotation processing

- Supported via kapt
- The only change in `build.gradle`:
  - `apt "com.google.dagger:dagger-compiler:2.7"`
  - `kapt "com.google.dagger:dagger-compiler:2.7"`

Performance and cost

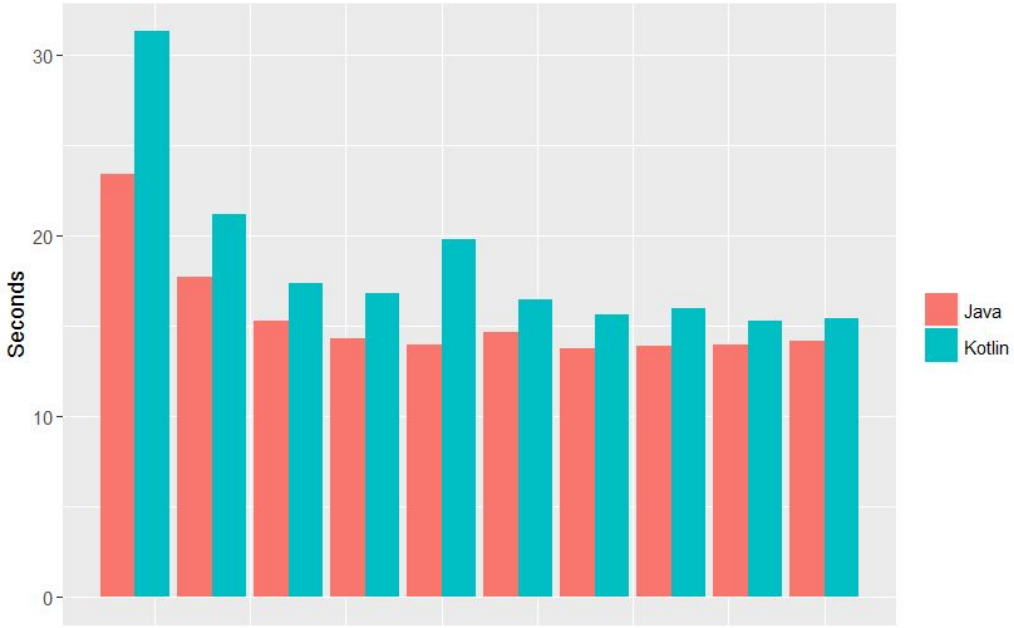
Performance

- Compiled to bytecode (like Java)
- No impact on performance
- Some Kotlin code faster
  - Lambdas that can be inlined
  - Built-in operations faster than DIY implementations

Build time

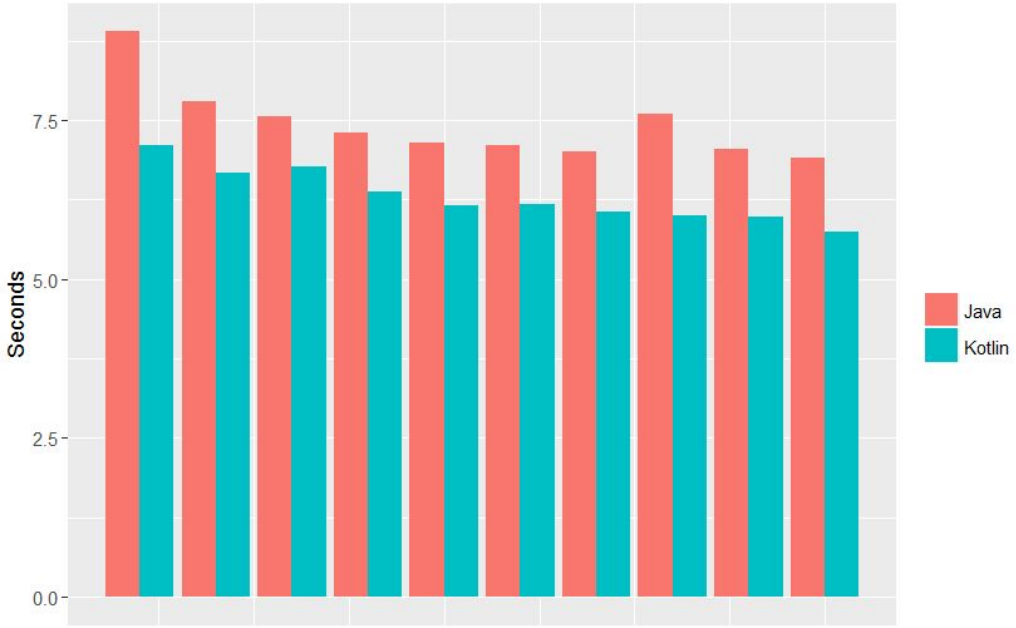
- Used to be a problem (in early releases)
- Much improved with incremental builds
- Keepsafe benchmarked compilation speed Kotlin vs Java
  - Link - goo.gl/WPs1Gx
- Configurations (Gradle daemon running):
  - Clean builds
  - Incremental build - isolated file change
  - Incremental build - core file change

Build time: clean



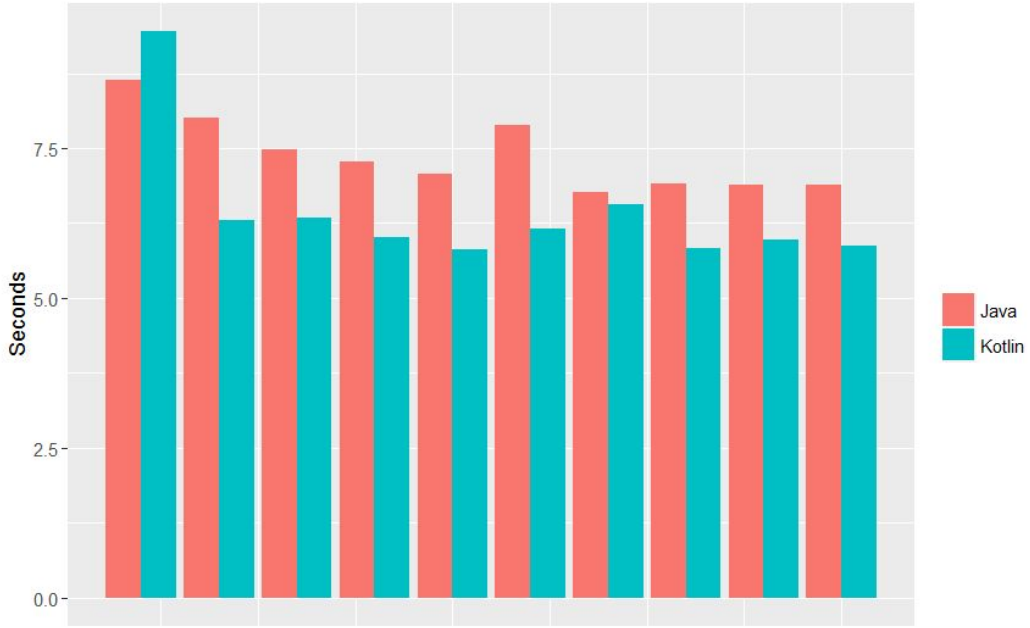
Courtesy of Keepsafe Engineering blog - [goo.gl/WPs1Gx](goo.gl/WPs1Gx)

Build time: incremental - isolated file change



Courtesy of Keepsafe Engineering blog - [goo.gl/WPs1Gx](goo.gl/WPs1Gx)

Build time: incremental - core file change



Courtesy of Keepsafe Engineering blog - [goo.gl/WPs1Gx](goo.gl/WPs1Gx)

Cost

- Kotlin Standard Library (1.0.4)
  - 5,723 methods
  - JAR size: 757 KB
  - DEX size: 1,012 KB
- For comparison:
  - Fresco (0.14.0) - 11,122 methods
  - Guava (19.0) - 15,076 methods
  - Google Play Service (5.0.77) - 20,298 methods

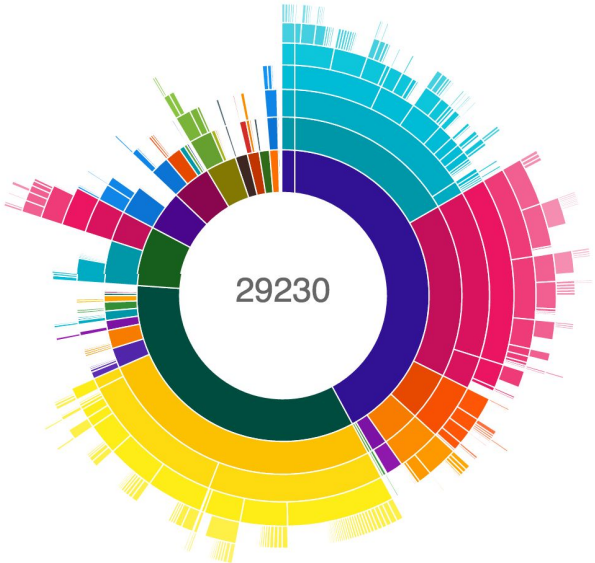
Case study

Case study

- Production app
  - Safedome
- Converted approx. 95% of the code to Kotlin
  - Kotlin 1.0.2 (early 2016)
- Enabled ProGuard
- Used Kotlin features (instead of straight conversion)

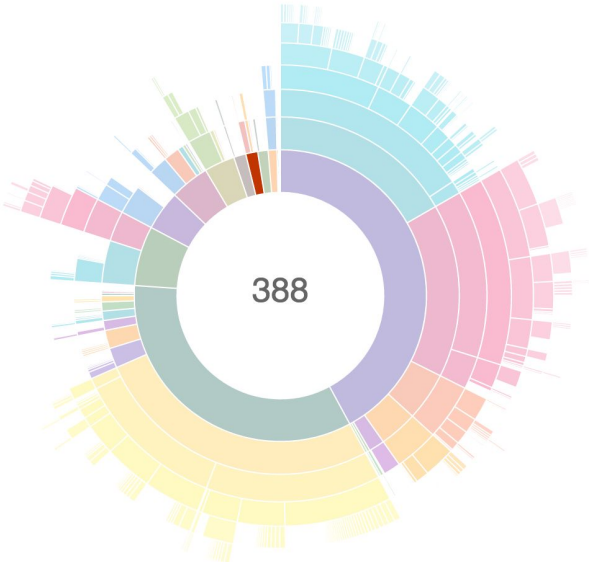
Method count

All methods ➜

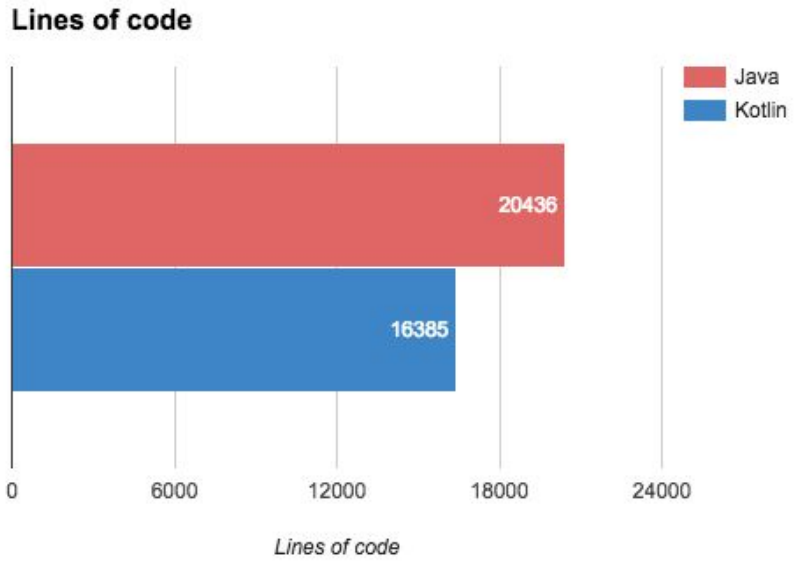


Method count

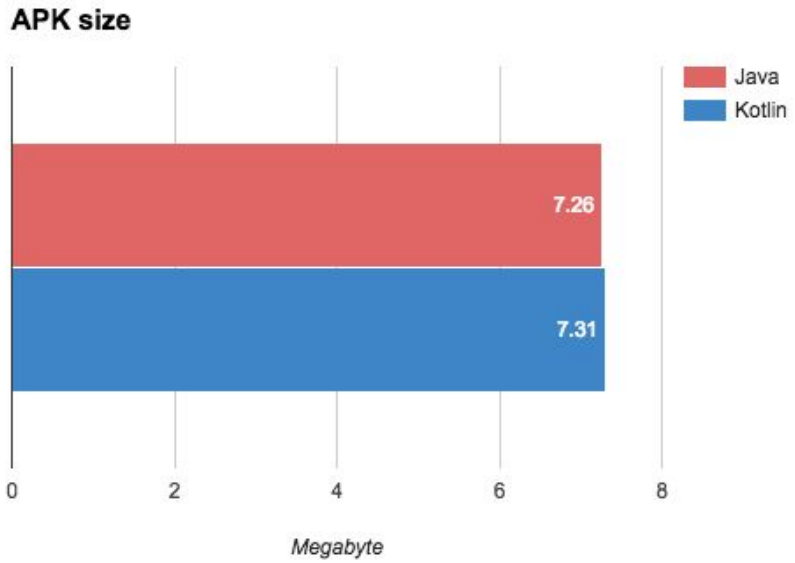
Kotlin methods ➜



Lines of code



APK size



Migration guide

Migration guide

- Simple process
  - Add Gradle dependencies (plugin, runtime, etc.)
  - Start writing `.kt` files instead of `.java` ones
- No need to migrate everything at once
  - Kotlin classes can co-exist with Java ones
- IntelliJ has a Java-to-Kotlin converter
  - Not perfect but good start
  - Works with pasted code

Migration fears

- Difficulty training developers
- Unsupported libraries

Were they founded?

Migration fears

- Difficulty training developers
- Unsupported libraries

Were they founded? **No**

Migration fears

- Difficulty training developers
  - Plenty of documentation
  - Desire to ditch Java motivates
- Unsupported libraries
  - Java libraries work just fine
  - Most current libraries have Kotlin support threads

Community adoption

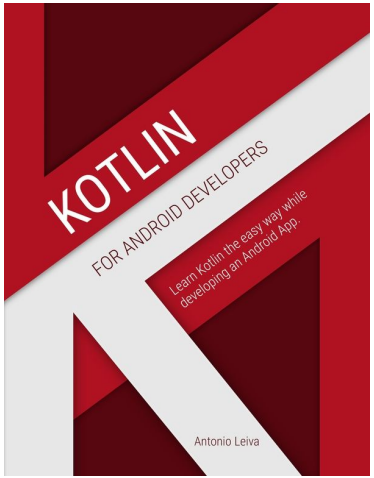
Community adoption

- Popular in the Android community
- Some companies using Kotlin in production:
  - Basecamp
  - NBC News Digital
  - Hootsuite
  - Prezi

Contributions

- Libraries
  - Spek, Wasabi, RxKotlin and many more
- Documentation
  - Books, articles, tutorials
- Other IDE support
  - Eclipse
  - NetBeans

Spek



eclipse

NetBeans

Gradle support

- Write scripts/plugins in Kotlin (since Gradle 3.0)
  - Note: Groovy not deprecated or removed... for now
- Works with Android plugin (since 2.2)
- Better IDE support and performance

Not just Android

- Kotlin is not limited to Android
- Just happens to be a good match
- Other applications
  - Back end: Spring, Vert.x, etc.
  - Front end: JavaScript
  - Any other Java applications

Current issues

Current issues

- Issue #1: Reflection
  - Requires `kotlin-reflect` import
  - Works fine if you need it
  - ...but it adds 8k methods!
- Solution:
  - Write files requiring reflection in Java
  - Example: Realm models

Current issues

- Issue #2: IntelliJ plugin stability
  - Plugin crashes sometimes
  - Doesn’t crash the whole IDE
- Solution:
  - Not a major annoyance
  - Only happens when doing something dodgy

Future releases

Future releases

- 1.0.x track
  - Bug fixes
  - Stability improvements
  - IDE support
- 1.1.x track
  - New features
  - Breaking changes (potentially)

Kotlin EAP 1.1

- Coroutines
- Type aliases
- Bound callable references
- Local delegation properties & inline properties
- Relaxed rules for sealed classes and data classes
- Scripting
- Java 7/8 support
- JavaScript

Kotlin EAP 1.1 (relevant to Android)

- Coroutines
- Type aliases
- Bound callable references
- Local delegation properties & inline properties
- Relaxed rules for sealed classes and data classes
- Scripting
- Java 7/8 support
- JavaScript

Summary

Summary

- Kotlin is a light, modern, compact language
- Compatible with Android
- No significant performance overhead
- Allows for gradual migration
- Becoming widely adopted
- In active development
- Ready for production

Thank you!

- Resources - gouline.net/talks
- Documentation - kotlinlang.org/docs/reference
- Kotlin Weekly - kotlinweekly.net

More Kotlin talks at **YOW! Connected 2016**:

- *“Anko - The Ultimate Ninja of Kotlin Libraries?”*
  - Speaker: Kai Koenig