PROGRAMMING LANGUAGE OF A MODERN DEVELOPER

KOTLIN

PAVOL RAJZÁK, ITERA SLOVAKIA

itera
MAKE A DIFFERENCE

About me

Software developer and technology enthusiast.

- **facebook**.com/pavol.rajzak
- sk.**linkedin**.com/in/rajzak
- **github**.com/rapasoft
- **blog**.rapasoft.eu
- http://**stackoverflow**.com/users/1471785/rapasoft

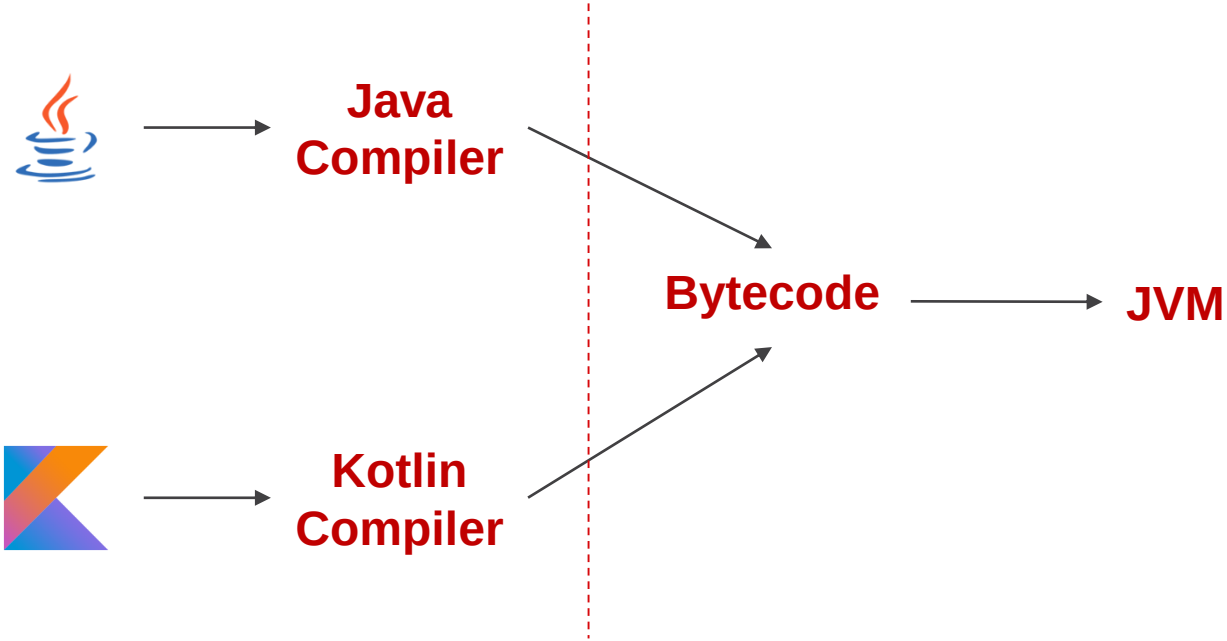
itera
MAKE A DIFFERENCE

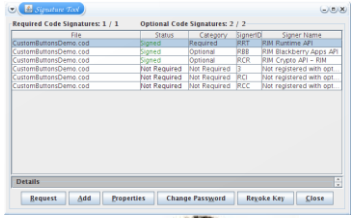
Content

- Motivations for learning yet another JVM language
- Kotlin – main selling points
- Direct comparison with Java syntax and some of the coolest features
- Integration with Java frameworks
- Is Kotlin production ready?

JVM Languages

Why learn JVM language





Why learn JVM language

- New Java releases postponed, features pushed away to future ones
- Process of adding new language features difficult
- Oracle's lack of interest? (e.g. Java EE Guardians)

- Released February 2016, adding new features (version 1.0.5 so far, 1.1 in spotlight)
- Community driven – sources on GitHub
- Powered by JetBrains (IntelliJ IDEA, ReSharper, TeamCity,…)

Kotlin – key selling points

- It’s free! (Open-source)
- Easy to follow syntax designed to make development a pleasure
- Modern language features
  - Type inferences, functional paradigms, null-safety, fluent APIs,…
- Tries to resolve and fix common pitfalls at compile time
- Java interoperability – seamless integration with existing frameworks
- Compatible with Java 1.6 runtimes (Android)
- Tooling and support

Kotlin vs. Java by examples

```java
private static List<Integer> filterSmallerThanHundred(List<Integer> intList) {
    List<Integer> output = new ArrayList<>();

    for (Integer x : intList) {
        if (x < 100) {
            output.add(x);
        }
    }

    return output;
}
```

```kotlin
intList.filter { it < 100 }
```

```java
intList.stream().filter(x -> x < 100).collect(Collectors.toList())
```

Kotlin vs. Java by examples

```java
public static Integer sum(List<Integer> list) {
    return list.stream()
            .reduce((i, acc) -> i + acc)
            .orElseThrow(RuntimeException::new);
}

public static void main(String[] args) {
    System.out.println(sum(null));
}
```

```
Exception in thread "main" java.lang.NullPointerException
        at Test.sum(Test.java:7)
        at Test.main(Test.java:11)
```

```fun
fun sum(list: List<Int>): Int {
    return list.sum()
}

fun main(args: Array<String>) {
    print(sum(null))
}
```

```
5  fun main(args: Array<String>) {
6      print(sum(null))
7  }
```

Null can not be a value of a non-null type List<Int>

Kotlin vs. Java by examples

…a Java class named Person with 4 fields, getters/setters and hashCode/equals/toString methods

```
data class Person(
        val firstName: String,
        val lastName: String,
        val occupation: String,
        val age: Int
)
```

Kotlin vs. Java by examples

Here be dragons!

```java
public class StringUtils {

    public static String appendJoke(String sentence) {
        return sentence + "... not!";
    }

    public static void main(String[] args) {
        System.out.println(
            appendJoke("I love this presentation")
        );
    }

}
```

```kotlin
fun String.joke() = this + "... not!";

fun main(args: Array<String>) {
    print("I am so excited about this presentation".joke())
}
```

What I like about Kotlin is that…

```
fun `This can be a nice descriptive name for a test`() = {}
```

…is a valid function name

```
fun booleanToEnglish(bool: Boolean) = if (bool) "Yes" else "No"
```

…if can be in-lined as an expression

```
val map = mapOf("a" to 1, "b" to 2, "c" to 3)
```

…map can be created easily

```
println("Map from above is $map")
```

…it can do string interpolation

Map from above is {a=1, b=2, c=3}

```
data class Range(val a: Int, val b: Int)
val range: Range = Range(0,1)
val (min, max) = range
```

…support destructors

And many more!

https://kotlinlang.org/docs/reference/idioms.html

itera
MAKE A DIFFERENCE

Java interoperability

- You can use any Java library
- Most frameworks and tools work without any problems (Maven, Spring, Hibernate…)
- There are caveats, though (null pointer checks, reflection, mocking frameworks)

```
fun main(args: Array<String>) {
    println("HelloWorld")
    System.out.println("Hello world!")
}
```

Documentation for out

< 1.8 >

java.lang.System

public static final java.io.PrintStream out

The "standard" output stream. This stream is already open and ready to accept output data. Typically this stream corresponds to display output or another output destination specified by the host environment or user.

For simple stand-alone Java applications, a typical way to write a line of

Java interoperability

Spring Boot - https://kotlinlang.org/docs/tutorials/spring-boot-restful.html

```
@RestController
class GreetingController {

    val counter = AtomicLong()

    @RequestMapping("/greeting")
    fun greeting(@RequestParam(value = "name", defaultValue = "World") name: String): Greeting {
        return Greeting(counter.incrementAndGet(), "Hello, $name")
    }
}
```

Is it production ready?

YES

…but

- IDE experience could be improved (refactoring, postfix completion, frameworks…)
- Data classes cannot be subtyped,… (planned for Kotlin 1.1)
- No `parallelStream()`-like alternative
- Kotlin-native frameworks still need some time to develop

Tip of the day: Start with small modules / refactoring when integrating Kotlin in project

Productive

Prevent NPEs

Intuitive

Java Compatible

Tooling and support

THANK YOU FOR YOUR ATTENTION

itera

MAKE A DIFFERENCE