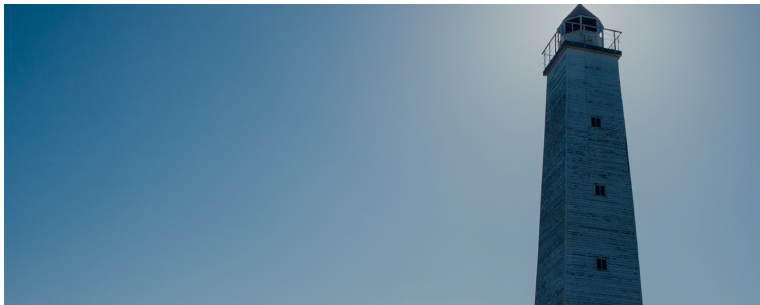


Kotlin 101

Christoph Pickl

Vienna, Austria – October 10th, 2016



A lighthouse on [Kotlin Island](#), Russia

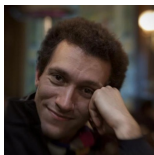
Introduction



- Statically typed, **hybrid** programming language for the **JVM**
- Fully **interoperable** with Java
- Runs on **Android** (generates 1.6 bytecode)
- Possibility to compile to **JavaScript**
- Focuses on **industry**, tooling and safety
- **Open source** compiler and tools (Apache 2 license)

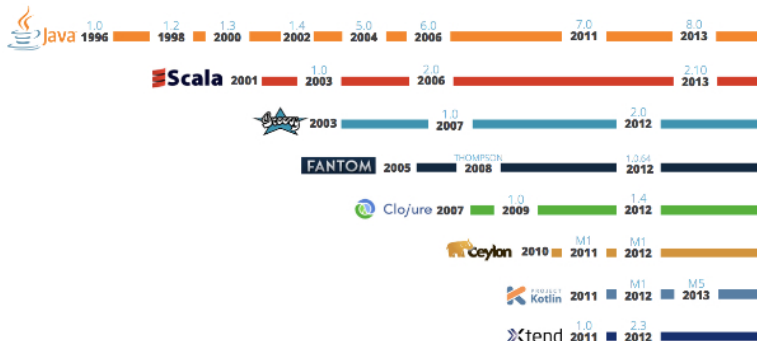


- Developed by **JetBrains** (IntelliJ, ReSharper, WebStorm, ...)
- Development already started **2010**
- Since then used in **projects** at JetBrains, Prezi and Expedia
- 2M+ LoC and 100+ contributors at **GitHub**
- **Version 1.0** released February, 2016



Andrey Breslav

Yet another JVM language?



Source: RebelLabs



“Most people talk about Java the language, and this may sound odd coming from me, but I could hardly care less. At the core of the Java ecosystem is the JVM.”

James Gosling,

Creator of the Java Programming Language (2011, TheServerSide)



Yet another JVM language!



- Well known company with good **tooling** support
- It got nothing new, just the **best of all** of them
- It's about the **ecosystem**, not the language
 - *Empirical Analysis of Programming Language Adoption* ([PDF](#))
- It's really, really easy to **learn**
 - *"Scala for ~~dummies~~ the masses"*

Language Features

We'll have a quick look at ...



- 1 Type inference
- 2 Declarations
- 3 Lambdas
- 4 Null handling
- 5 Smart casts
- 6 Properties
- 7 Extension methods
- 8 Named & default arguments
- 9 Data classes
- 10 Collection API



```
1 // mutable variable of type Int
2 var a = 42
3
4 // immutable value of type String
5 val b = "foobar"
6
7 // explicit type declaration
8 val c: Double = 13.37
9
10 // whatever the return type is
11 val d = someFunction()
```



```
1 // global function with explicit return type
2 fun add1(x: Int, y: Int): Int {
3     return x + y
4 }
5
6 // compact one-line syntax
7 fun add2(x: Int, y: Int) = x + y
8
9 // possibility to mark it as infix function
10 infix fun Int.add3(y: Int) = this + y
11 // enables us to write:
12 20.add3(22)
```



```
1 // single ctor initializing a property
2 class Greeter(private val prefix: String) {
3
4     // string interpolation, no concatenation
5     fun greet(name: String) =
6         "$prefix ${name}!"
7 }
```

Object declaration



```
1 // an object is a singleton done properly
2 object Highlander : SwordFighter {
3     // mandatory 'override' keyword
4     override fun slash() { ... }
5 }
6
7 // looks like a nasty static invocation
8 Highlander.slash()
9
10 // we only want to kill real sword fighters
11 fun killHim(fighter: SwordFighter) { ... }
12
13 // pass an (the!) instance reference
14 killHim(Highlander)
```



Kotlin is well known for being a **concise language**:

```
1 // no surrounding class necessary
2 fun main(args: Array<String>) {
3
4     // no 'new' keyword necessary
5     val greeter = Greeter("Hello")
6
7     // no 'System.out' reference necessary
8     println(greeter.greet("sIT"))
9 }
```




```
1 // with java 8:
2 Stream.of(1, 2, 3).filter(i -> i % 2 == 0)
3   .collect(Collectors.toList());
4
5 // with kotlin:
6 listOf(1, 2, 3).filter { i -> i % 2 == 0 }
7
8 // or even shorter (groovy style):
9 listOf(1, 2, 3).filter { it % 2 == 0 }
```

PS: If the last argument of a function is a function, it can be outside the paranthesis; if there are no other arguments, you can even omit them.

Map function



```
1 fun <T, R> map(list: List<T>,
2     transform: (T) -> R): List<R> {
3     val result = arrayListOf<R>()
4     for (item in list)
5         result.add(transform(item))
6     return result
7 }
8
9 // invoke the function and pass a lambda
10 map(listOf(1, 2, 3), { it * 2 })
```





```
1 // nullable types for better compile checks
2 val maybe: String? = ...
3
4 maybe.length // COMPILE ERROR!!!
5 maybe?.length // type is Int?
6 maybe?.length ?: -1 // type is Int
7 maybe!!.length // i dont f*cking care!
8
9 if (maybe != null) {
10     maybe.length // smart cast to String
11 }
```



Java

```
Object x = ...;
if (x instanceof A) {
    a = (A) x;
    a.foo();
} else if (x instanceof B) {
    B b = (B) x;
    b.bar();
} else {
    throw new Exception("Sad panda!");
}
```

Kotlin

```
val x: Any = ...
when (x) {
    is A -> x.foo()
    is B -> x.bar()
    else -> throw Exception("Sad panda!")
    // there are no checked exceptions :)
}
```

Pojo without Properties



```
1 public class Account {
2
3     private String id;
4     private int amount;
5
6     public Account(String id, int amount) {
7         this.id = id;
8         this.amount = amount;
9     }
10
11     public String getId() {
12         return id;
13     }
14     public void setId(String id) {
15         this.id = id;
16     }
17     public int getAmount() {
18         return amount;
19     }
20     public void setAmount(int amount) {
21         this.amount = amount;
22     }
23 }
```



```
1 class Account {  
2     var id: String? = null  
3     var amount: Int = 0  
4 }  
5  
6 // usage sample:  
7 val account = Account()  
8 account.id = "asdf"  
9 account.amount = 100  
10 // invoke getter/setter from java as usual
```



```
1 class Account {  
2     var amount = 0  
3     get() = field // backing field  
4     set(value) {  
5         if (value < 0) throw SomeException()  
6         field = value  
7     }  
8     // ...  
9 }
```




```
1 // primary ctor initializing properties
2 class Account(
3     var id: String,
4     var amount: Int
5 )
6 // no need for curly braces
```

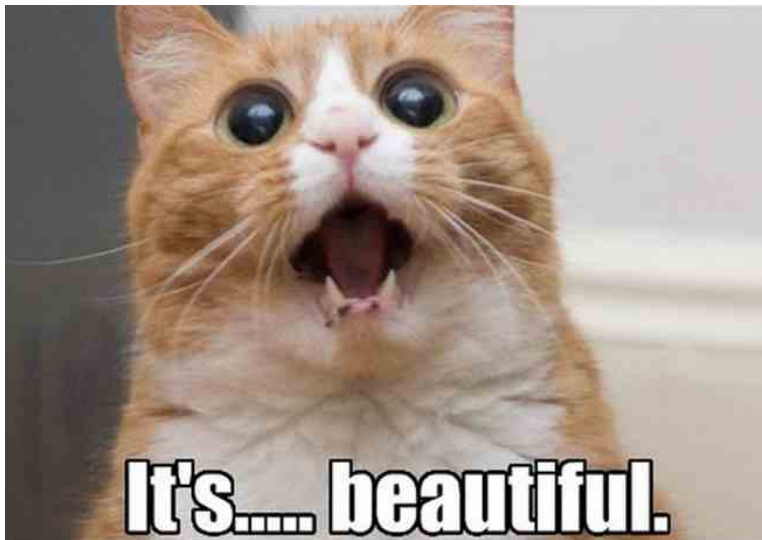


```
1 String agent = "7";  
2  
3 // we are used to call methods like this:  
4 agent.length();  
5  
6 // but as String is final we do this:  
7 StringUtil.pad(agent, 3, "0");
```

How many Util classes are out there? A gazillion maybe?!



```
1 fun String.pad(length: Int,
2     symbol: String): String {
3     // "this" refers to current string
4     ...
5 }
6
7 val agent = "7"
8 agent.pad(3, "0") // auto-completion FTW!!!
9
10 // nullable receiver transformation
11 fun Money?.toDTO() =
12     if (this == null) null else MoneyDTO(this)
```





- Known from C#, Smalltalk, Ruby and others
- Add methods to yet compiled (final) classes
- No syntactic **difference** between calling real vs extension
- Actually simply creates **static methods** in background
- Problem of **delocalization** solved, auto-completion works
- *Extension methods certainly are not object-oriented!*



```
1 fun greet(  
2     name: String,  
3     prefix: String = "Hello")  
4     = "$prefix, ${name}!"  
5  
6 greet("sIT", "Hi") => "Hi, sIT!"  
7  
8 greet("sIT") // => "Hello, sIT!"
```

No more overloaded methods and delegation necessary!



```
1 fun greet(  
2     name: String,  
3     prefix: String = "Hello",  
4     suffix: String = "!")  
5 { ... }  
6  
7 // specify the argument name and  
8 // skip the "prefix" param (use default)  
9 greet(name = "sIT", suffix = ".")
```



Java

```
public class Person {
    private final String name;
    private final int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    public boolean equals(Object other) {
        ...
    }
    public int hashCode() {
        ...
    }
    public String toString() {
        ...
    }
}
```

Kotlin

```
data class Person(
    val name: String,
    val age: Int
)

// same as with lombok but better ;)

// primary ctor deals with getters
// and field initialisation

// generates equals/hashCode/toString
// and a copy function (no builders!)
```




- Collections designed properly! (**im/mutability** as in Scala)
- Create a map: `mapOf("a" to 1, "b" to 2)`
- Access a map: `map["a"]`
- `size` property for collections and arrays
- Many, many useful (functional) **extension** methods ...
 - Guava R.I.P., you served well



```
1 // val list = listOf(1, 2, 3)
2 val list = mutableListOf(1, 2, 3)
3 list.add(42) // would not compile!
4 list.requireNotNulls()
5 list.first() // => 1
6 list.firstOrNull { it > 10 } ?: -1
7 if (list.none { it > 10 })
8     println("no big ones")
9 list.toList() // returns immutable copy
10
11 mapOf(1 to "a", 2 to "b", 3 to "a")
12     .values.distinct().toHashSet()
```



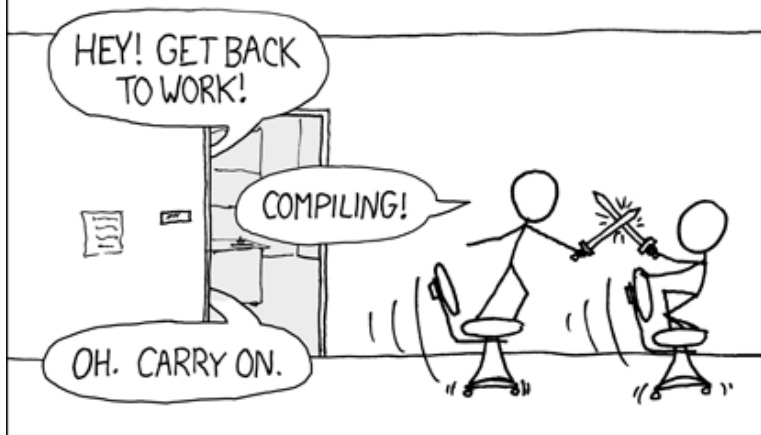
- Delegation
- String handling
- Sealed classes
- Pattern matching
- Reified generics
- Operator overloading
- `if`, `try` and `when` as expressions
- `==` as equals
- Range expressions
- Rename imports
- Backticks for escaping
- ...and much much more ...

Epilog



- Still pretty **young**, but things are getting better
 - Some language features missing
 - Unsatisfying tool support (static code analysis)
- Final by default is good, but bad for **mocking**
- `kotlinc` is significantly **slower** than `javac`

THE #1 PROGRAMMER EXCUSE
FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."





- Kotlin feels like **Java 2.0** (*Java on steroids*)
- It's backed by a **company** and still **open source**
- Pretty awesome **community** work
- Version **1.0** just has been released (begin 2016)
- **Roadmap** promises lots of improvements



- Try Kotlin online
<http://try.kotlinlang.org>
- Learn by Koans
<https://kotlinlang.org/docs/tutorials/koans.html>
- Ask, Discuss, Complain
<https://kotlinlang.slack.com>
- Presentation Sources
<https://github.com/christophickl/kotlin101slides>
- Kotlin Vienna Usergroup
<http://www.meetup.com/Kotlin-Vienna>

Vienna, Austria
Founded May 10, 2016

[About us...](#)[+ Invite friends](#)

Members 43

Past Meetups 1

Our calendar 

Organizers:



Christoph Leiter,
Christoph Pickl

[✉ Contact](#)

We're about:

Java · Open Source ·
Software Development ·
Programming Languages ·
Android · Computer
programming · Java
Virtual Machine · Kotlin

Welcome, Members!

[+ Schedule a new Meetup](#)

Upcoming Past Calendar



There are no upcoming Meetups

You can schedule one!

[Schedule a Meetup](#)

Recent Meetups

3 days ago · 6:15 PM

A first unofficial top secret meetup



10 Members |  | 1 Photo

Let's get to know each other at our first meetup. We can share our Kotlin experiences in a relaxed atmosphere and talk about the

What's new



 NEW COMMENT

Christoph Pickl
commented on A
first unofficial top secret
meetup



2 days ago

 NEW PHOTO COMMENT

Christoph Pickl
commented on a
photo



2 days ago

 NEW PHOTO



Christoph Pickl added a
new photo to Meetup
People Pictures

2 days ago

 NEW COMMENT

Siegfried Goeschl
commented on A
first unofficial top secret
meetup





Most have seen [presentation](#) by Dr. Russel Winder