otlin 101

Christoph Pickl

Vienna, Austria – June 13th, 2018

A lighthouse on Kotlin Island, Russia

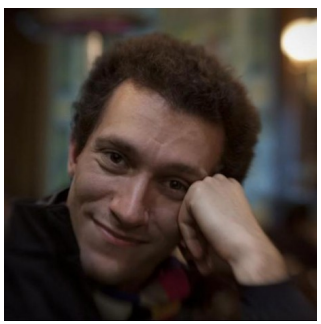
Introduction

Language fundamentals

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

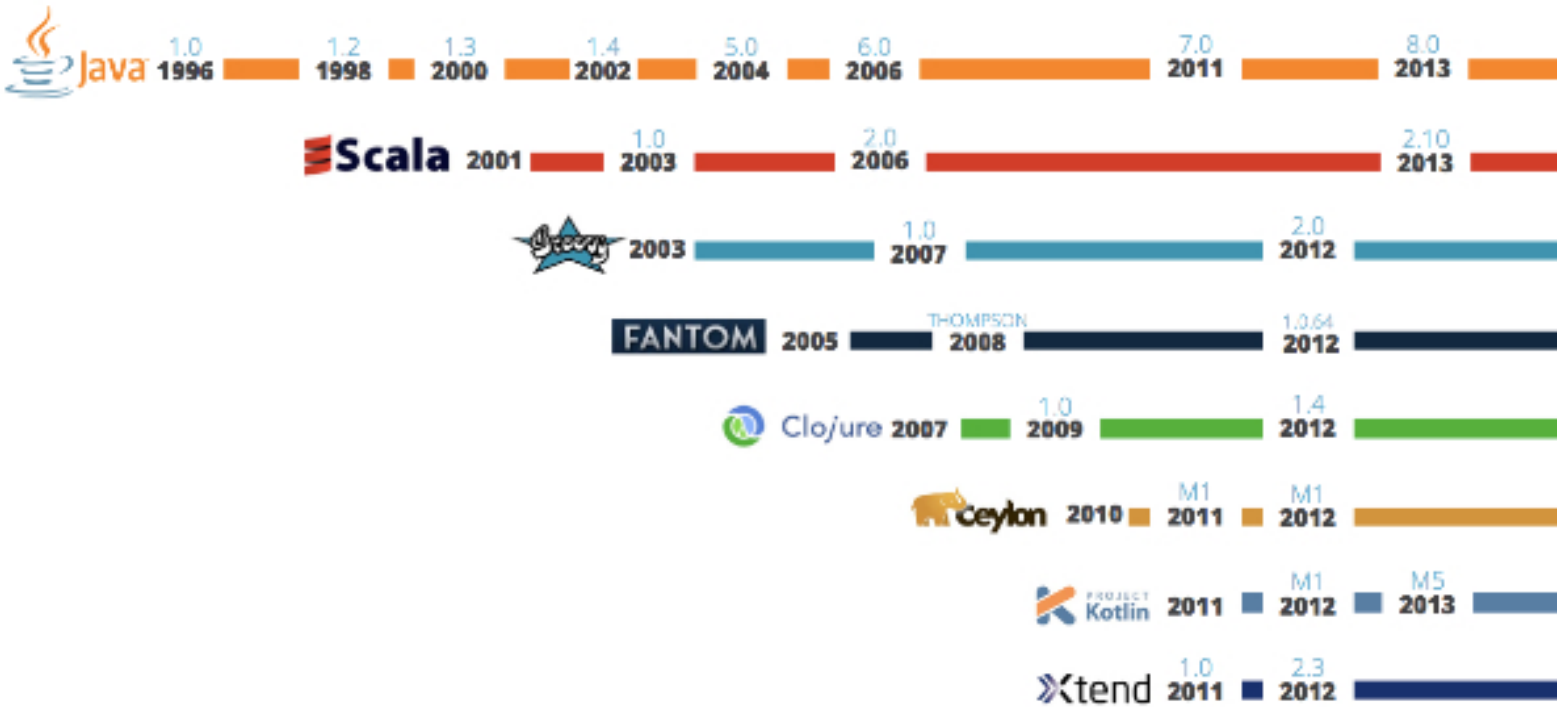
Historic abstract

- Developed by **JetBrains** (IntelliJ, ReSharper, WebStorm, . . .)
- Development already started back in **2010**
- 2M+ LoC and 200+ contributers at **GitHub**, +30k repos
- Version 1.0 released February, 2016



Andrey Breslav

Yet another JVM language?



Source: RebelLabs

Java is dead, long live Java

"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)


U SAY JAVA IZ
NO COOL ANYMOAR?!
memegenerator.net

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. Arguments
9. Data classes
10. Collection API

Type inference

```kotlin
// mutable variable of type Int
var a = 42

// immutable value of type String
val b = "foobar"

// explicit type declaration
val c: Double = 13.37

// whatever the return type is
val d = someFunction()
```

Function declaration

```
1  // global function with explicit return type
2  fun add1(x: Int, y: Int): Int {
3    return x + y
4  }
5
6  // compact single expression syntax
7  fun add2(x: Int, y: Int) = x + y
8
9  // possiblity to mark it as infix function
10 infix fun Int.add3(y: Int) = this + y
11
12 // enables you to write:
13 20 add3 22
```

Class declaration

```
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)
```

A simple application

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("ERSTE"))
9 }
```

Lambdas compared

```
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

```
fun <T, R> map(list: List<T>,
    transform: (T) -> R): List<R> {
  val result = arrayListOf<R>()
  for (item in list)
    result.add(transform(item))
  return result
}

// invoke the function and pass a lambda
map(listOf(1, 2, 3), { it * 2 })
```


I DON'T ALWAYS CODE IN JAVA
BUT WHEN I DO, IT'S A NULLPOINTEREXCEPTION

Null handling

```
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 }
```

Smart casts

Java

```java
final 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

```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

```java
public class Account {

    private String id;
    private int amount;

    public Account(String id, int amount) {
        this.id = id;
        this.amount = amount;
    }

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public int getAmount() {
        return amount;
    }
    public void setAmount(int amount) {
        this.amount = amount;
    }
}
```

Pojo with Properties

```
class Account {
    var id: String? = null
    var amount: Int = 0
}

// usage sample:
val account = Account()
account.id = "asdf"
account.amount = 100
// invoke getter/setter from java as usual
```

Explicit Properties

```
class Account {
    var amount = 0
        get() = field // backing field
        set(value) {
            if (value < 0) throw SomeException()
                field = value
        }
        // ...
}
```

Compact Properties

```
// primary ctor initializing properties
class Account(
    var id: String,
    var amount: Int
)
// no need for curly braces
```

Extension methods – The Problem

```java
String agent = "7";

// we are used to call methods like this:
agent.length();

// but as String is final we do this:
StringUtil.pad(agent, 3, "0");
```

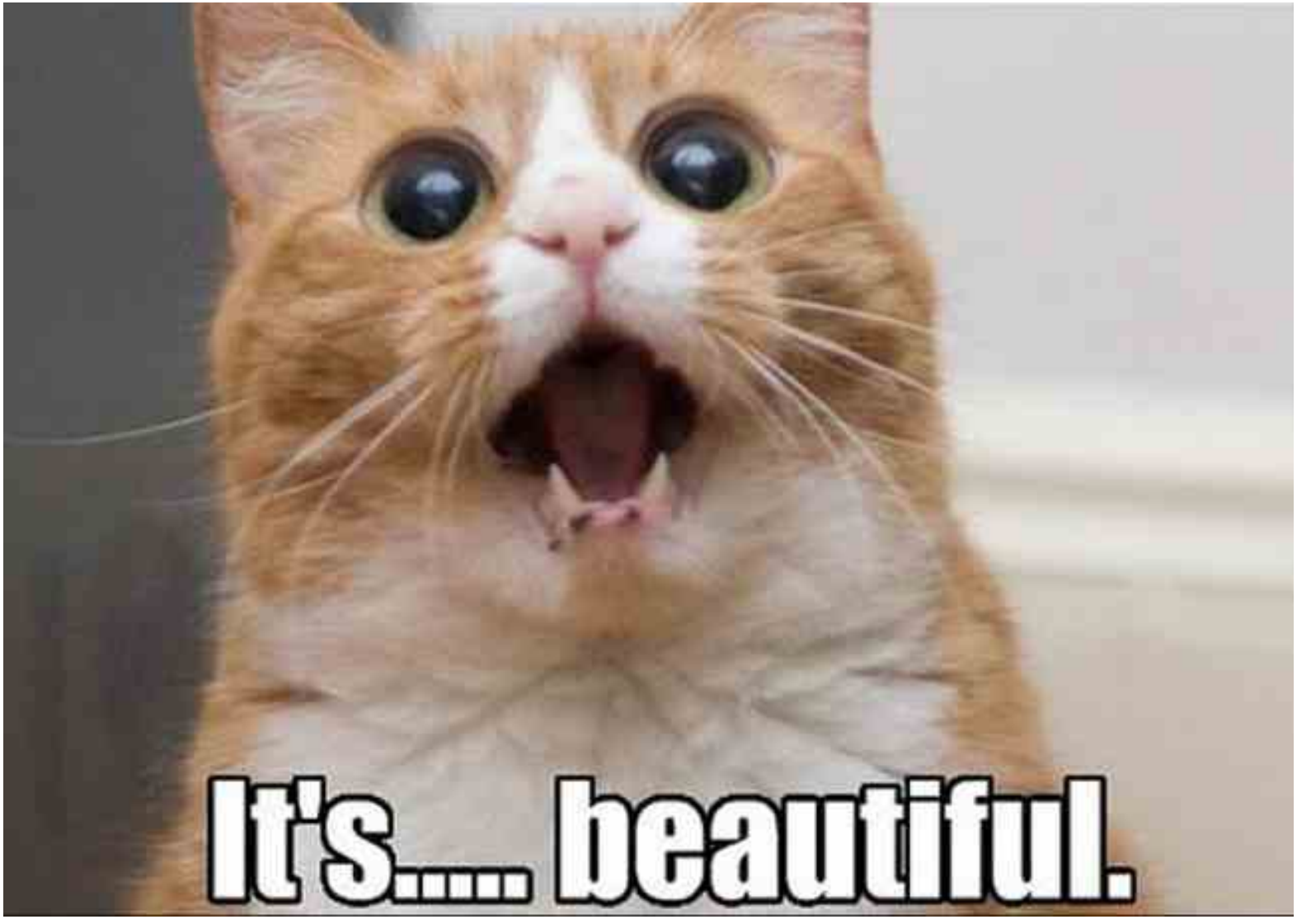
How many Util classes are out there? A gazillion maybe?!

Extension methods – The Solution

```kotlin
fun String.pad(length: Int,
    symbol: String): String {
  // "this" refers to current string
  ...
}

val agent = "7"
agent.pad(3, "0") // auto-completion FTW!!!

// nullable receiver transformation
fun Money?.toDTO() =
  if (this == null) null else MoneyDTO(this)
```


It's..... beautiful.

Extension Methods – Remarks

- 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!*

Default arguments

```
fun greet(
    name: String,
    prefix: String = "Hello")
  = "$prefix, ${name}!"

greet("ERSTE", "Hi") => "Hi, ERSTE!"

greet("ERSTE") // => "Hello, ERSTE!"
```

No more overloaded methods and delegation necessary!

Named arguments

```
fun greet(
    name: String,
    prefix: String = "Hello",
    suffix: String = "!")
  { ... }

// specify the argument name and
// skip the "prefix" param (use default)
greet(name = "ERSTE", suffix = ".")
```

Data classes

Java

```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

```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!)
```

Collection API

- 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
  - And so did Lombok

Collection API – Samples

```
// val list = listOf(1, 2, 3)
val list = mutableListOf(1, 2, 3)
list.add(42) // would not compile!
list.requireNoNulls()
list.first() // => 1
list.firstOrNull { it > 10 } ?: -1
if (list.none { it > 10 })
  println("no big ones")
list.toTypedArray()

mapOf(1 to "a", 2 to "b", 3 to "a")
  .values.distinct().toHashSet()
```

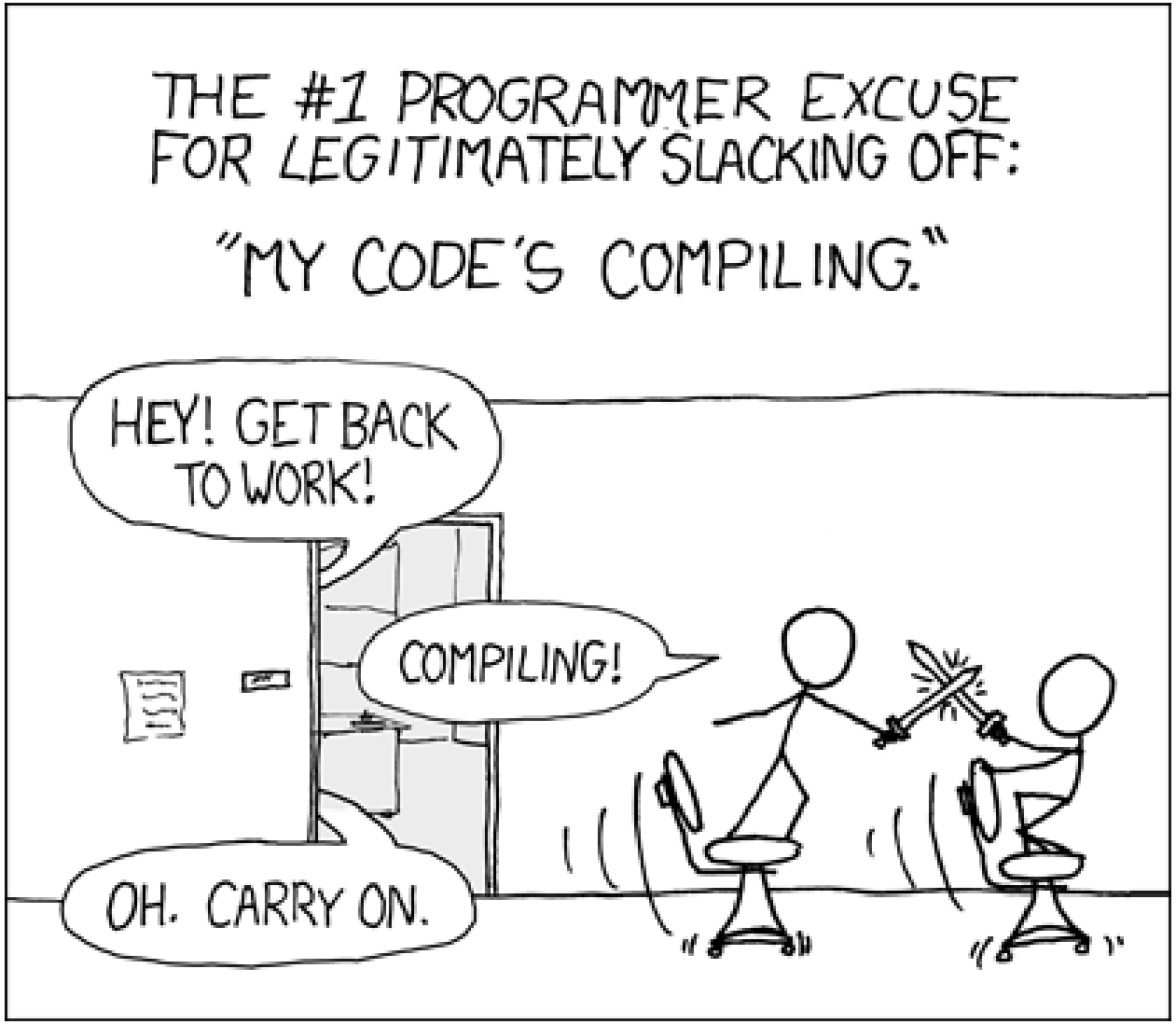
Further Features

- Delegation (lazy)
- String handling (lots of extension functions)
- Sealed classes (algebraic datastructures anyone?)
- Pattern matching (or something like it)
- Reified generics (fake it until you make it)
- Operator overloading (to some extent, at least for BigOnes)
- `if`, `try` and `when` as expressions
- == as equals (for improved readability)
- Range expressions
- Rename imports
- Backticks for escaping
- . . . and much much more . . .

Epilog

Some Downsides

- Still pretty **young**, but things are getting better
  - Some language features missing but coming soon
  - Unsatisfying tool support (static code analysis)
- Some frameworks just don't adhere to Kotlin's philosophy
  - Final by default is good, but ...
- `kotlinc` is significantly **slower** than `javac`


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

Source: xkcd

Summary

- Kotlin feels like **Java 2.0** (*Java on steroids*)
- It's backed by a **company** and still **open source**
- Pretty awesome **community** work
- Version **1.2** has been released (end 2017)
- **Roadmap** promises lots of improvements

Further reading

- Try Kotlin online
  http://try.kotlinlang.org
- Ask, Discuss, Complain
  https://kotlinlang.slack.com
- Presentation Sources
  https://github.com/christophpickl/kotlin101slides
- Kotlin Vienna Usergroup
  http://www.meetup.com/Kotlin-Vienna

Kotlin Vienna

```
fun main(args: Array<String>) {
    listOf("Kotlin", "JVM", "Android")
        .forEach { println("Hello $it!") }
}
```

Home | Members | Sponsors | Photos | Pages | Discussions | More | Group tools | My profile

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

Now for some Kotlin Koans