My Language of Choice

Experiences with Kotlin

Who am I

- Christian Dräger
- Software Developer from Berlin

christian-draeger

_chris_draeger_

christian-draeger

What's Kotlin

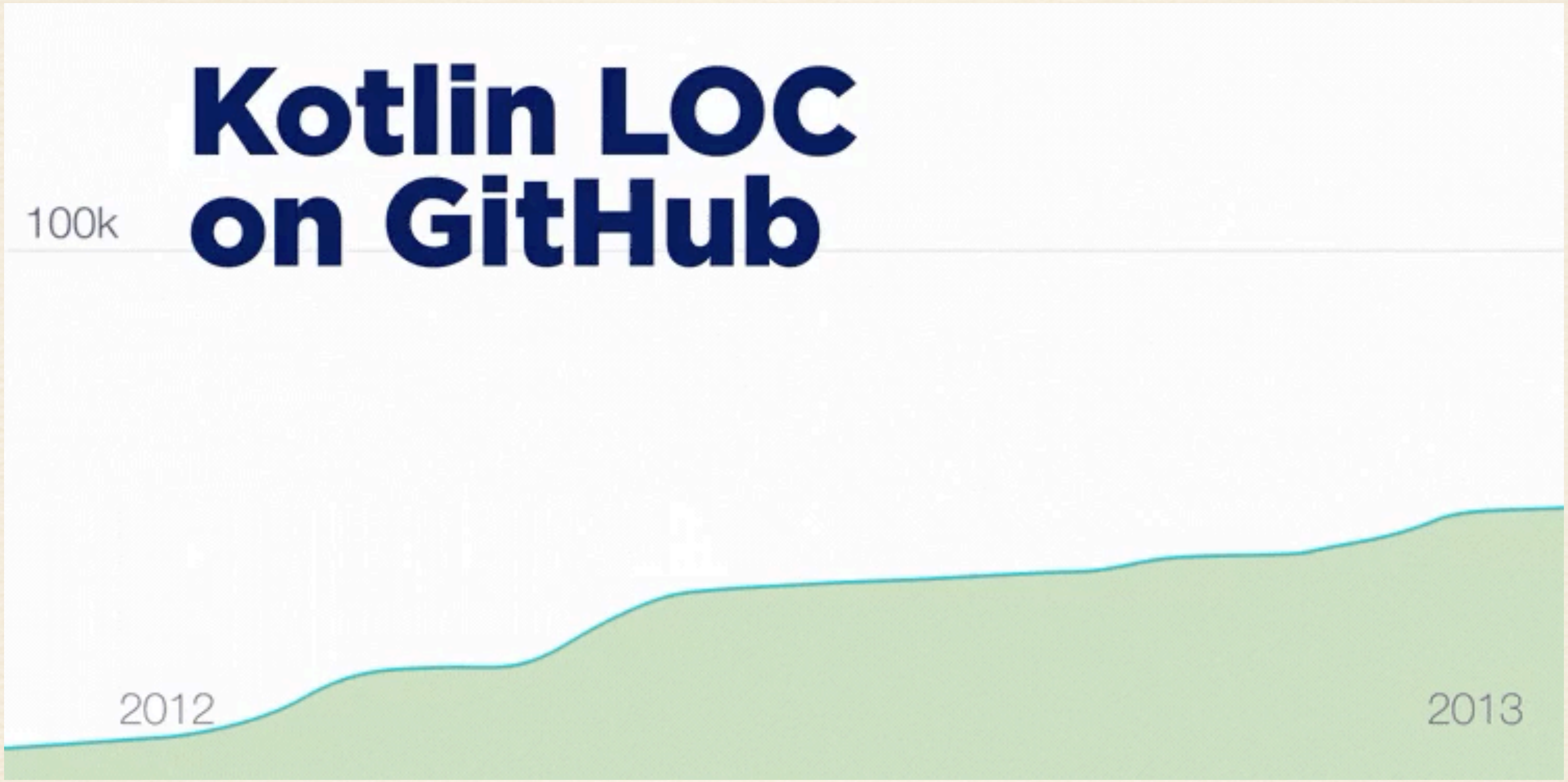
- Statically Typed
- Open Source
- sponsored by Jetbrains
- the better Java?

Who is using Kotlin?

- compiles to different platforms
  - JVM, Android, JS, C (native), iOS
- runs like a charm on the JVM and Android
- dedicated Kotlin support in Spring Framework 5.0+

May 2017 (Google I/O):

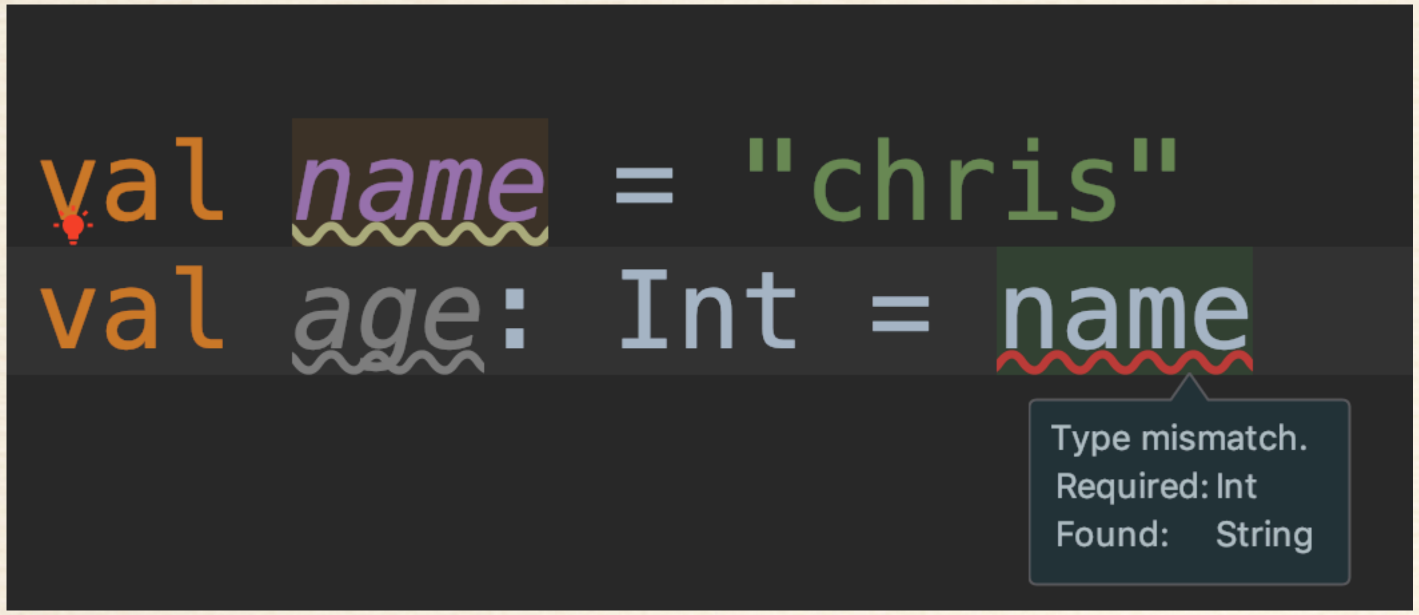
„Today the Android team is excited to announce that we are officially adding support for the Kotlin programming language. Kotlin is a brilliantly designed, mature language that we believe will make Android development faster and more fun.“


Kotlin LOC
on GitHub
100k
2012
2013

Why it Rocks?

- feels like Java as it always should be
- low learning curve without surprises
- very well documented with huge community
- a lot of nice language features
- Java interop (on the JVM)

type inference



```
val name = "chris"
val age: Int = name
```

```
Type mismatch.
Required: Int
Found:    String
```

```
val foo = listOf("foo", "bar")
val bar = mutableListOf("foo", "bar")
val map = mapOf("foo" to 1, "bar" to 2)
```

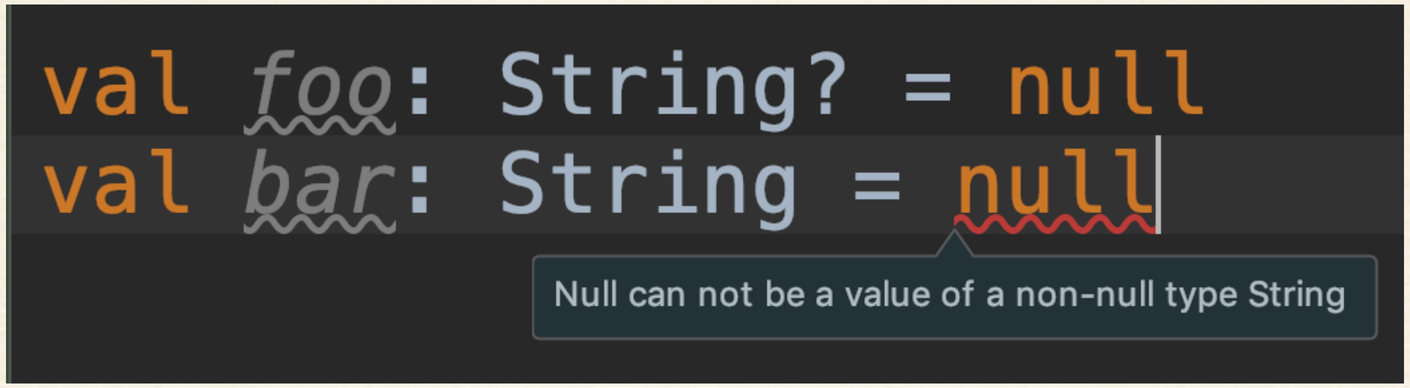
- reduces boilerplate - increases readability

type alias

```
typealias PersonIndex = Map<String, Person>
```

- handy for functional types or a types with type parameters which is used multiple times in a codebase
- will behave exactly the same like Map<String, Person> but is explizit and increases readability

null-safety



```
val foo: String? = null
val bar: String = null
```

Null can not be a value of a non-null type String

- Types are not nullable
- if needed you have to make it explizit by adding ? at end of Type

String handling

```
val multiline: String =
        """
        foo,
        bar
        """.trimIndent()
```

```
val name = "chris"
println("hey I'm $name")
```

extension functions

```kotlin
// with block body syntax
fun String.codeFreeze(): String {
   return "codefreeze-$this"
}

"Chris".codeFreeze()
// will add „codefreeze-“ at the beginning
```

```kotlin
// with expression body syntax
fun String.codeFreeze(): String = "codefreeze-$this"
```

data classes

```kotlin
data class Person(
        var name: String,
        val email: String
)
```

- generating
  - getters
  - setters (if var)
  - equals / hashcode
  - copy
  - toString

    "Person(name=John, email=foo@bar.tld)"
- reduces boilerplate

default arguments

```kotlin
class MyClass {
  @Test
  fun `can print Person with default arguments`() {
    println(Person("john").toString())
  }
}

data class Person(
    val name: String = "chris",
    val email: String = "foo@bar.tld"
)
```

will print "Person(name=john, email=foo@bar.tld)"

named arguments

```kotlin
class MyClass {
   val person = Person(email = "bla@blub.de", name = "john")
}

data class Person(
     val name: String,
     val email: String
)
```

- builder pattern out-of-the-box 👍
- increase readability

a lot more nice features and facts

- a lot more intuitive and easy to use stream api compared to Java
- Coroutines
- Kotlin DSL
- Perfect IntelliJ Support (not surprisingly)
- uses existing and mature ecosystem of Java
  - buildtools: e.g. Maven, Gradle
  - Can use Java Libs out-of-the-box
- easy to get started, especially if you did Java or Swift before

testing

- this deserves an extra session to be honest.
- in short, all testing libs you from Java will work
- there are a lot of nice dedicated kotlin testing libs popping up atm

- to dig deeper i highly recommend this nice Talk I saw at the Kotlin Conf 2018 in Amsterdam:
  - https://youtu.be/RX_g65J14H0

puzzlers

- every language has it weaknesses
- you can find some funny puzzlers over here:
    - https://github.com/angryziber/kotlin-puzzlers
    - the community is constantly fixing these things

experiences / strategies to get started

- just add kotlin compiler plugin and kotlin-stdlib to your existing Java Project
  - convert a Java class to Kotlin
  - run all tests - build should still be green
    - generate trust by doing baby-steps
- write your tests in kotlin for the beginning
- …

let’s hack