Kotlin: Al Fresco

Brandt Gigle

10.24.2018

Photo by Jasmin Schreiber on Unsplash

Who am I

Java

Android

IoT

Kotlin

Who are you

Photo by Annie Spratt on Unsplash

Outline

- What is Kotlin
- Language features
- What is missing
- Tips
- Q&A

Past

- Started in 2011
- Open-sourced in 2012
- v1.0 LTS in 2016

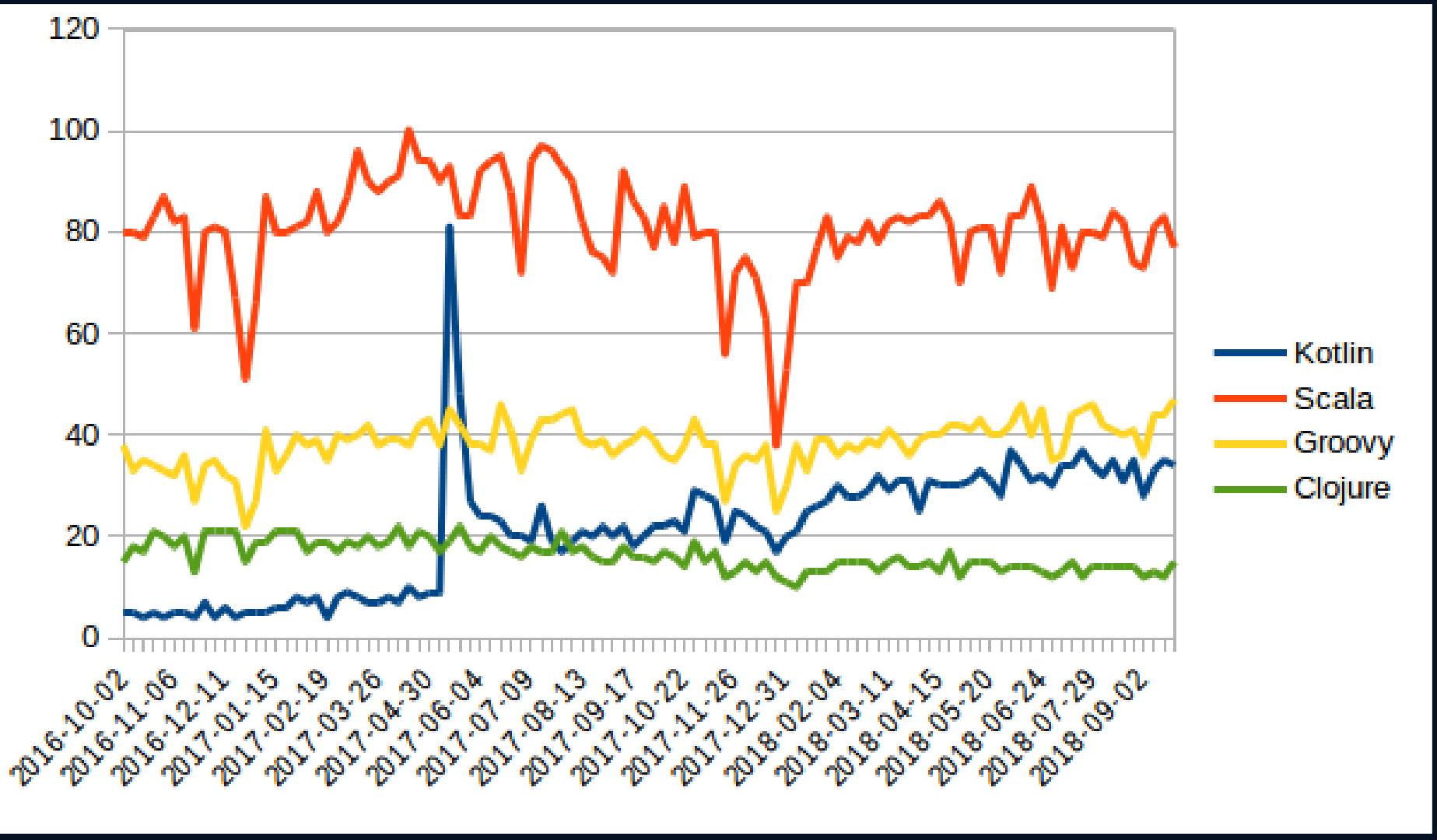
Present

- v1.3 RC in 2018
- kotlinc: Java 6 and 8
- kotlin2js: ECMAScript 5.1
- kotlin/native: iOS, MacOS, Android, Windows, Linux, WebAssembly

Kotlin in Philly

Photo by Kelly Kiernan on Unsplash

Future



Data source: Google Trends

Language features

- Classes & methods
- Data classes
- Objects
- Mutability
- Null safety
- Functions

Follow along

Browser-based editor at try.kotlinlang.org

Reference code at gist.github.com/bgigle

Classes in Java

```
public final class Hostess {
    public final String name;

    public Hostess() {
        this("Elizabeth");
    }

    public Hostess(String name) {
        this.name = name;
    }

    public boolean sendInvites(List<String> guestList) {
        System.out.println("Sent!");
        return true;
    }
}
```

Classes in Kotlin

Defining

```
class Hostess(val name: String = "Elizabeth") {
    fun sendInvites(guestList: List<String>): Boolean {
        println("Sent!")
        return true
    }
}
```

Using

```
val elizabethHostess = Hostess()

val hyacinthHostess = Hostess("Hyacinth")
hyacinthHostess.sendInvites(listOf("Daisy", "Onslow"))
```

Data classes

Defining

```
data class Salad(
    val isFresh: Boolean = true,
    val ingredients: Set<String> = emptySet())
```

Using

```
val delectableSalad = Salad(true, setOf("Lettuce", "Tomato"))

val scrumptiousSalad = Salad(ingredients = setOf("Kale", "Spam"))

val horrificSalad = scrumptiousSalad.copy(isFresh = false)

// menu = "Salad(isFresh=false, ingredients=[Kale, Spam])"
val menu = horrificSalad.toString()

val (delectableFreshness, delectableIngredients) = delectableSalad
```

Objects

Defining

```
object ChefUtil {
    fun makeSalad(salad: Salad) {
        for(ingredient in salad.ingredients) {
            println("Adding some $ingredient")
        }
        println("Bon Appetit!")
    }
}
```

Using

```
val finestSalad = Salad(true, setOf("Cheese", "Bun", "Hamburger"))
ChefUtil.makeSalad(finestSalad)
```

Companion objects

Defining

```
typealias Size = Pair<Int, Int>

class Blanket(val location: String){
    companion object {
        val DEFAULT_COLOR = "vanilla"

        val DEFAULT_SIZE = Size(4, 6)

        fun getInstance(): Blanket = Blanket("down by the river")
    }
}
```

Using

```
val bestColor = Blanket.DEFAULT_COLOR

val blanket = Blanket.getInstance()
```

Mutability

```
// flyingBlanket is read-only variable, i.e. final
val flyingBlanket = Blanket("flying through the stratosphere")
flyingBlanket = Blanket("in outer space") // does not compile

// securityBlanket is read-write variable
var securityBlanket = Blanket("in a baby crib")
securityBlanket = Blanket("in Fort Knox") // compiles
```

Warmup round

```
DinnerParty.prepare()
```

Null Safety

Photo by Andersen Jensen on Unsplash

Type system

```
// ambrosiaSalad is non-nullable
var ambrosiaSalad: Salad
ambrosiaSalad = null // does not compile

// glasswortSalad is nullable
var glasswortSalad: Salad?
glasswortSalad = null // compiles
```

Safe calls

Java

```
@Nullable
Guest guest;

final Boolean isLate = ((guest != null) &&
    (guest.getWienermobile() != null) &&
    (guest.getWienermobile().getEngine() != null))
    ? (!guest.getWienermobile().getEngine().isRunning()) : null;
```

Kotlin

```
val guest: Guest?

val isLate = guest?.wienermobile?.engine?.isRunning()?.not()
```

Elvis operator

Java

```java
@Nullable
String reason;

String excuse = (reason != null) ? reason : "My fish has the flu";
```

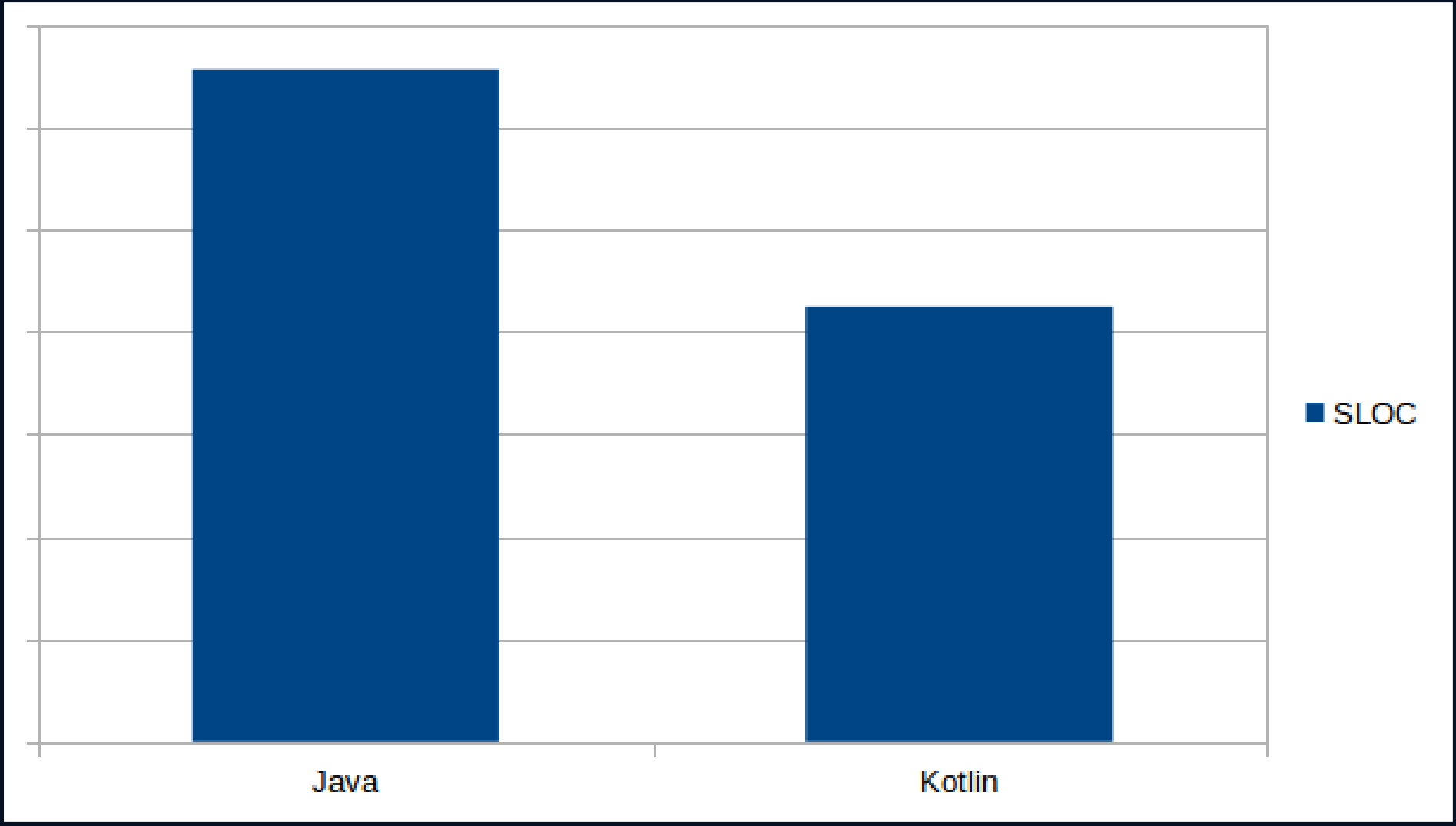
Kotlin

```kotlin
val reason: String?

val excuse = reason ?: "I'm allergic to spam"

val rationalization = reason ?: throw IllegalArgumentException()
```

Terseness



Data source: Chariot Solutions

Smart casting

```
val mysteryObj: Any?

if(mysteryObj is LocalDate) {
    println("${Period.between(LocalDate.now(), mysteryObj).days} days")
}

// can throw a form of class cast exception
var mysteryGuest = mysteryObj as Guest

// won't throw an exception, but 'salad' is now type Salad?
var mysterySalad = mysteryObj as? Salad
```

when function

```
val enigmaObj: Any?

val result = when(enigmaObj) {
    is Hostess -> { enigmaObj.name }
    in 41..43 -> { "The answer to life, the universe, and everything" }
    true, 1, "spam" -> { "It's an $enigmaObj" }
    guessAnObject() -> { "Great guess!" }
    else -> { "The mystery continues..." }
}
```

Functions

Photo by Susan Holt Simpson on Unsplash

Extension functions

Defining

```
fun LocalDate.reschedule(days: Long): LocalDate {
    return this.plusDays(days)
}
```

Using

```
val dinnerDate = LocalDate.parse("2018-10-24")
val betterDate = dinnerDate.reschedule(7)

// invite = "2018-10-31"
val invite = betterDate.toString()
```

Higher-order functions

Defining

```
fun obtainRsvp(contact: (Int) -> Salad): Salad {
    println("Looking for an RSVP")
    return contact.invoke(5)
}

fun callSheridan(count: Int): Salad {
    for (i in 0..count) {
        println("Phone call #$i")
    }
    return Salad()
}
```

Using

```
var firstRsvp = obtainRsvp(::callSheridan)

var secondRsvp = obtainRsvp { count: Int ->
    println("Emailing $count times")
    Salad()
}
```

Collections & lambdas

```
val rsvpOrders: List<Salad?>

val shoppingList = rsvpOrders.filterNotNull()
    .parallelStream()
    .flatMap { it.ingredients.stream() }
    .filter { it != "bacon bits" }
    .distinct()
    .sorted { i1, i2 -> i1.compareTo(i2) }
    .collect(Collectors.toList())

val lastIngredient = shoppingList.lastOrNull()

val hasNoAvocado = shoppingList.none { it == "avocado" }
```

Final round

```
DinnerParty.commence()
```

What is missing

Checked exceptions

```
public abstract boolean tossSalad() throws IOException;
```

Primitive types

Static members

Ternary operators

```
boolean result = isSunShining ? sunbathe() : weep()
```

```
val result = if (isSunShining) sunbathe() else weep()
```

Tips

- Think small
- Start with your data model
- Learn the converter. Love the converter.
- Watch those compile times

Further exploration

Ponder over Kotlin Koans

Investigate Kotlin for Android developers

Peruse Kotlin in Action

Chat on the kotlinlang Slack channels

Chat on the #kotlin freenode IRC channel

Q&A

Enjoy!

Photo by Hannan Busing on Unsplash

Questions, comments, feedback

Email me at bgigle@chariotsolutions.com

Message me @Brandt Gigle on kotlinlang Slack channels

I am also at linkedin.com/in/brandt-gigle/

Slides at chariotsolutions.com/presentation/kotlin-al-fresco-brandt-gigle/