

Groovy, Java, and Kotlin

Functional Programming

Contact Info

Ken Kousen

Kousen IT, Inc.

ken.kousen@kousenit.com

<http://www.kousenit.com>

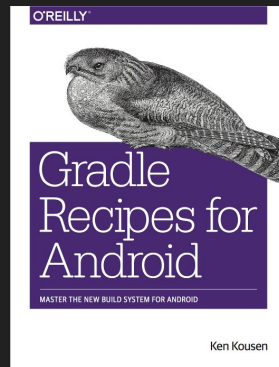
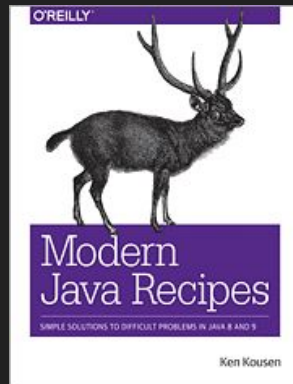
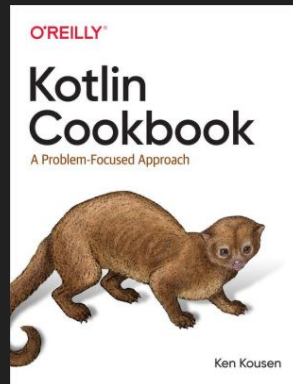
<http://kousenit.org> (blog)

[@kenkousen](https://twitter.com/kenkousen) (twitter)

<https://kenkousen.substack.com> (newsletter)

All demos at:

https://github.com/kousen/java_groovy_kotlin



Kotlin Certified Training Partner

- [Certified by JetBrains](#) for Kotlin



Kotlin Certified Training Partner

- [Certified by JetBrains](#) for **Kotlin**
- (Would do the same thing for **Groovy** if a program existed :)

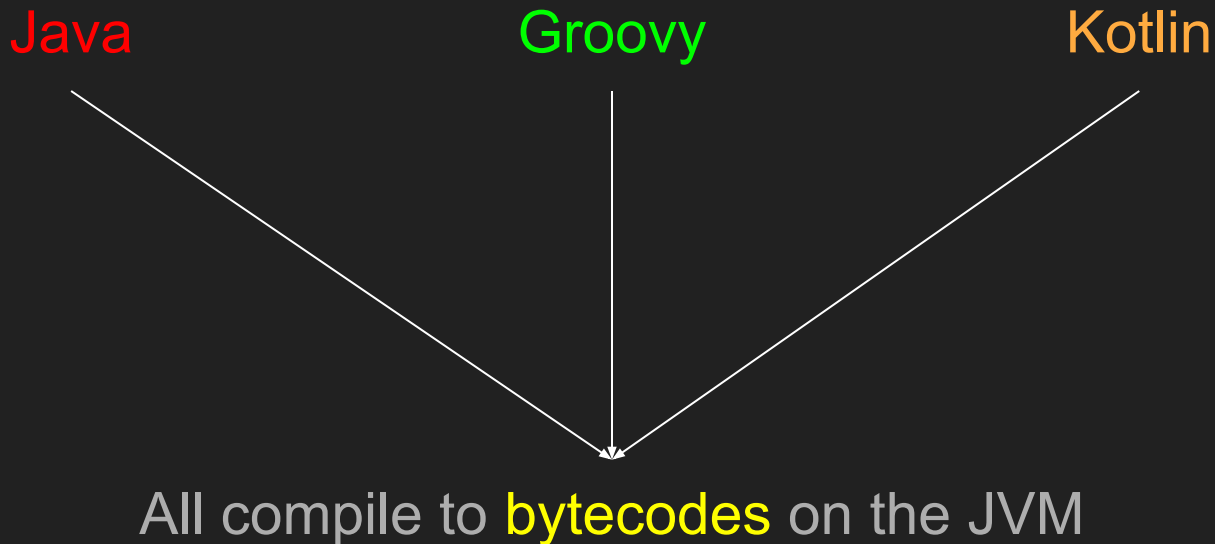


Kotlin Certified Training Partner

- [Certified by JetBrains](#) for **Kotlin**
- (Would do the same thing for **Groovy** if a program existed :)
- (Ditto for **Java**)



Java, Groovy, and Kotlin

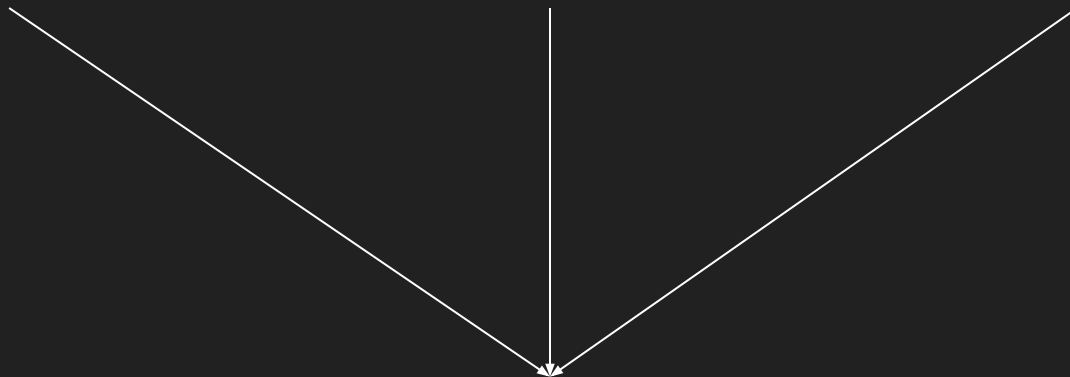


Java, Groovy, and Kotlin

Java

Groovy

Kotlin



None are **functional** languages

Java, Groovy, and Kotlin

Java

Groovy

Kotlin



```
graph TD; Java --> Conclusion; Groovy --> Conclusion; Kotlin --> Conclusion;
```

They are OO languages with functional features

Java

Just had its 25 year anniversary

Managed by Oracle, but with many implementations

OpenJDK, Azul, Amazon Corretto, SAP, ...

Functional features added in version 1.8

Current LTS version is 11

Current version is 14

Java 15 will be released in Sept 2020

Groovy

Apache project

<https://groovy-lang.org>

Syntax very similar to Java

Easy for Java devs to learn

Most Groovy projects are actually combined Java + Groovy projects

Current version is 3.0.*

Grails, Spock, Gradle, Ratpack, Geb, Griffon, ...

Kotlin

Managed by JetBrains

<https://kotlinlang.org>

Compiles to JVM, or generates JS
Mobile and native available

Definitive language for Android
Google develops Android libs "Kotlin first"
Gradle Kotlin DSL also available

Current version is 1.4.*

Lambda Expressions

Java lambda expressions

Lambda Expressions

Java lambda expressions

Assigned to Single Abstract Method interfaces

Lambda Expressions

Java lambda expressions

Assigned to Single Abstract Method interfaces

Parameter types inferred from context

Lambda Expressions

Java lambda expressions

Assigned to **S**ingle **A**bstract **M**ethod **interfaces**

Parameter types inferred from context

Can **access** local variables but **not modify** them

final or "effectively final"

Lambda Expressions

Groovy closures

Lambda Expressions

Groovy closures

Are instances of the class `groovy.lang.Closure`

Lambda Expressions

Groovy closures

Are instances of the class `groovy.lang.Closure`

Have a `delegate` property for resolving members

Lambda Expressions

Groovy closures

Are instances of the class `groovy.lang.Closure`

Have a `delegate` property for resolving members

Can `access` local variables and `modify` them

Lambda Expressions

Kotlin lambda expressions

Lambda Expressions

Kotlin lambda expressions

Are expressions of the form

$$(\text{InputType1}, \text{InputType2}, \dots) \rightarrow \text{OutputType}$$

Lambda Expressions

Kotlin lambda expressions

Are expressions of the form

$(\text{InputType1}, \text{InputType2}, \dots) \rightarrow \text{OutputType}$

Have a **receiver** for resolving members

Lambda Expressions

Kotlin lambda expressions

Are expressions of the form

$(\text{InputType1}, \text{InputType2}, \dots) \rightarrow \text{OutputType}$

Have a **receiver** for resolving members

Can **access** local variables and **modify** them

(Are really closures)

Functional Programming

Lambda, Method References, and Streams

LambdaDemo.java

MapFilterReduce.java

PrimeChecker.java

StreamsDemo.java

Functional Programming

Groovy closures

Many methods added to collections

collect, findAll, inject vs map, filter, reduce

closures.groovy

map_filter_reduce.groovy

PrimeCheckerGroovy.groovy

Kotlin Functions

Return type shown after signature

```
fun sum(a: Int, b: Int) : Int {  
    return a + b  
}
```

Single-expression functions:

```
fun sum(a: Int, b: Int) = a + b
```

Return type inferred

primes.kt

Lazy Streams

Java Streams are lazy

Only process as much data as needed

Groovy and Kotlin collections are not

But Groovy can use Java streams and Kotlin has sequences

LazyStreams.java

LazyStreamsGroovy.groovy

LazyStreamsSpec.groovy

Kotlin Sequences

Methods like `map`, `filter`, `reduce`, and `fold` are added to collections

The "`asSequence()`" method converts collection to sequence

Like Java streams

Evaluated element at a time

No data processed unless there is a terminal expression

`lazySequences.kt`

`sequenceFunction.kt`

map, filter, reduce

Java:

```
int total = myNums.stream()  
    .filter(n → n % 3 == 0)  
    .map(n → n * 2)  
    .reduce(0, (acc, val) → acc + val);
```

```
total = myNums.stream()  
    .filter(n → n % 3 == 0)  
    .mapToInt(n → n * 2)    // map to IntStream  
    .sum();
```

map, filter, reduce

Groovy:

```
int total = nums.findAll { it % 3 == 0 }  
    .collect { it * 2 }          // unfortunate name  
    .inject(0) { acc, val -> acc + val }
```

```
total = nums.findAll { it % 3 == 0 }  
    .collect { it * 2 }  
    .sum() // duck typing
```

map, filter, reduce

Kotlin:

```
int total = nums.filter { it % 3 == 0 }  
    .map { it * 2 }  
    .fold(0) { acc, val -> acc + val }
```

```
total = nums.filter { it % 3 == 0 }  
    .map { it * 2 }  
    .sum()
```

map, filter, reduce

Kotlin:

```
int total = nums.asSequence()  
    .filter { it % 3 == 0 }  
    .map { it * 2 }  
    .sum()
```

Sequence works like Java stream:

each num through entire pipeline before next
requires terminal operation to process values

POGOs vs POJOs

Plain Old **Groovy** Objects

private attributes, public methods, public class

map-based constructor

generated getters and setters

@Canonical → toString, equals, hashCode, tuple ctor

POJOs vs POGOs

Sorting streams with POJOs/POGOs

Golfer.java, SortGolfers.java

GroovyGolfer.groovy, sort_groovy_golfers.groovy

GroovyGolferCS.groovy

Kotlin Data Classes

Classes defined using the keyword "data"

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

(That's the entire class)

Data classes have:

- generated getters and setters
- `toString`, `equals`, `hashCode`
- `copy()` method
- `componentN()` methods for destructuring

Beyond map/filter/reduce

Closure Composition

Memoization, Tail Recursion

Currying

Closure Composition

Java:

default methods in `Consumer`, `Predicate`, ...
`CombineLambdas.java`

Groovy:

right-shift operator
`composition.groovy`

Kotlin:

write your own
`composition.kt`

Tail Recursion

Groovy:

`@TailRecursive` // AST transformation

Kotlin:

`tailrec` // keyword

Beyond map/filter/reduce

AST transformations:

`AnnotatedFunctions.groovy`

`UseAnnotatedFunctions.groovy`

`AnnotatedFunctionsTest.groovy`

Libraries

Library to add functional capabilities to **Kotlin**

Arrow, <https://arrow-kt.io/>

For **Groovy**, every **Java** library can be used right away

Metaprogramming

Groovy can add methods to a class at runtime or compile time

pirate.groovy

Metaprogramming

Kotlin allows for **extension** functions added at compile time

palindrome.kt

primes.kt

Cat Pictures

Flickr RESTful (sort of) web service

<https://www.flickr.com/services/api/flickr.photos.search.html>

Parsing JSON

SwingBuilder

Can use Java parallel streams

cat_pictures.groovy

Summary

Can use **Groovy**, **Kotlin**, and **Java** together
Use each for what it does well

Java functional capabilities are limited

Groovy goes beyond them and is quite mature
builders, parsers, metaprogramming, traits, AST transformations

Kotlin does too, but has its own quirks

All demos at: https://github.com/kousen/java_groovy_kotlin