“Why Kotlin?”

Value & Adoption

CHRISTOPHER.ROBERTS@KOHLS.COM

Staff Software Engineer

Agenda

- **Kotlin@Kohls**
- “Why Kotlin?”
- Features - Demo
- Adoption Suggestions
- Open Discussion

Kotlin@Kohls

(Note: Captured from slack: #developer-kotlin-community)

- Limited Use
- Licensing
- Compatibility
- Efficiency
- Future Plans

Kotlin@Kohls - Limited Use

(Note: Captured from slack: #developer-kotlin-community)

- Android development
- Personal projects

Kotlin@Kohls - Licensing

(Note: Captured from slack: #developer-kotlin-community)

- Apache 2.0
- Avoids potential legal issues

Kotlin@Kohls - Compatibility

(Note: Captured from slack: #developer-kotlin-community)

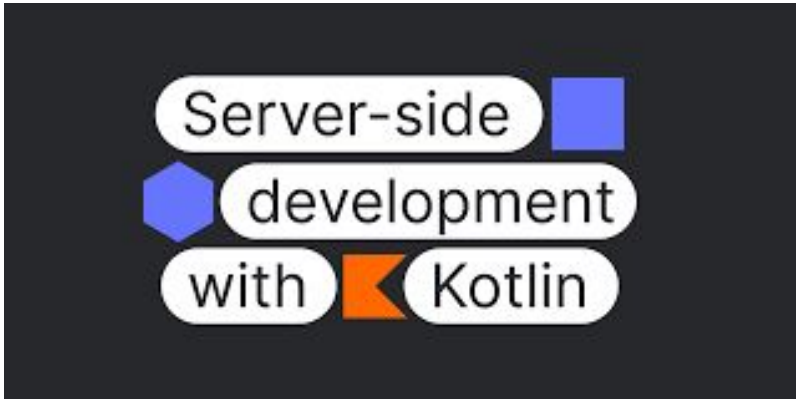
- Fully Java compatible
- Smooth transition

Kotlin@Kohls - Efficiency

(Note: Captured from slack: #developer-kotlin-community)

- Write less code
- Reduce cognitive load

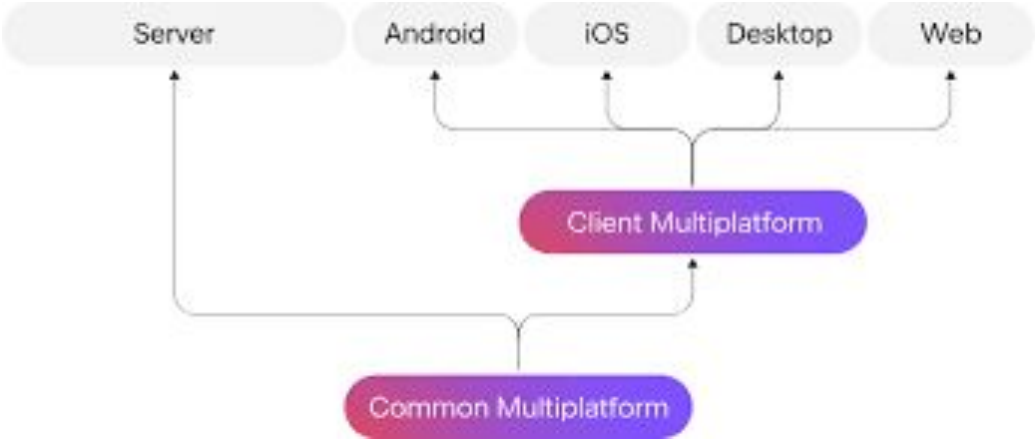
Kotlin@Kohls - Future Plans



(Note: Captured from slack: #developer-kotlin-community)

Migrating…

“In-store Server Services” to cloud



Agenda

- Kotlin@Kohls
- **“Why Kotlin?”**
- Features - Demo
- Adoption Suggestions
- Open Discussion

“Why Kotlin?” - First What is **Kotlin**

- **Modern** Java **alternative**
- **Concise** syntax design
- **Interoperable** with Java

“Why Kotlin?” - Personal Analogy

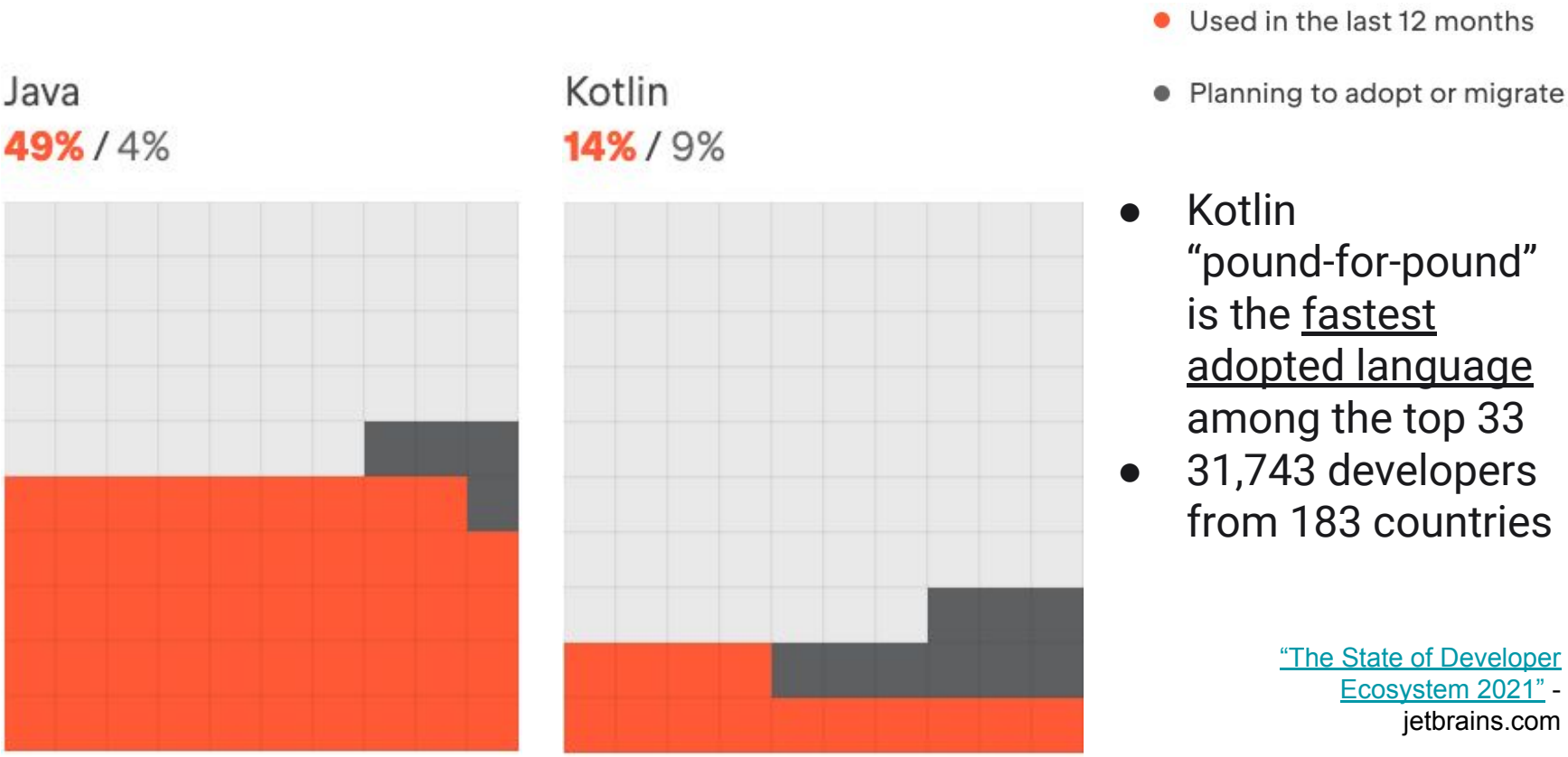
“Why Kotlin?” - Personal Analogy

- Java is like childhood

“Why Kotlin?” - Personal Analogy

- Java is like childhood
- Kotlin is like adolescence

“Why Kotlin?” - Fastest Adopted Language ‘21



- Kotlin “pound-for-pound” is the fastest adopted language among the top 33
- 31,743 developers from 183 countries

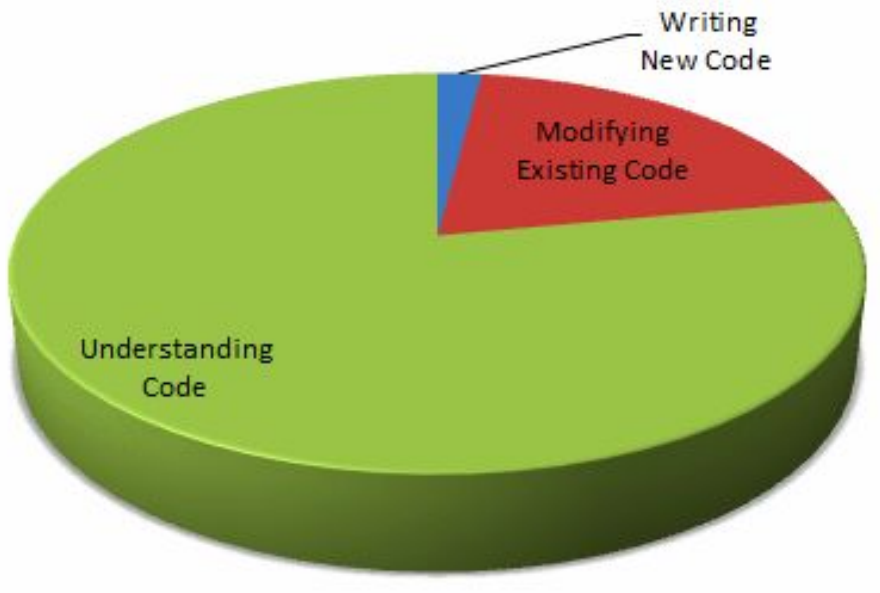
“The State of Developer Ecosystem 2021” - jetbrains.com

“Why Kotlin?” - Increased Productivity

"Indeed, the ratio of time spent ***reading versus writing is well over 10 to 1****. We are constantly reading old code as part of the effort to write new code. ...[Therefore,]* ***making it easy to read makes it easier to write.****"*

Robert C. Martin (“Uncle Bob”)

"When Understanding means Rewriting"



“Why Kotlin?” - Answer?

- In light of this evidence

Why Kotlin?

“Why Kotlin?” - Answer?

- In light of this evidence
- Turn question onto itself

Why Kotlin?

“Why Kotlin?” - Answer?

- In light of this evidence
- Turn question onto itself

Why NOT Kotlin?

Agenda

- Kotlin@Kohls
- “Why Kotlin?”
- **Features - Demo**
- Adoption Suggestions
- Open Discussion

Features - Basic “Hello, World!”

```
fun main() { println("Hello, World!") }
```

Features - Variables and Type Inference

```
val greeting: String = "Hello"
var name = "World"
println("$greeting, $name!")
```

Features - String Templates

```
println("$greeting, ${name.toUpperCase()}!")
```

Features - Functions

```
fun greet(greeting: String, name: String) = "$greeting, $name!"
println(greet(greeting, name))
```

Features - Default and Named Arguments

```
fun greet(greeting: String = "Hi", name: String = "there") =
"$greeting, $name!"
println(greet(name = "Kotlin"))
```

Features - Data Classes

```
data class Person(val firstName: String, val lastName: String=“Smith”)
val person = Person("John", "Doe")
println(greet(name = person.firstName))
```

Features - Null Safety

```
var nullableName: String? = null
println(nullableName?.toUpperCase() ?: "Name is null")
```

Features - Extension Functions

```
fun String.exclaim() = "$this!"
println("Hello".exclaim())
```

Features - Lambdas

```
val shout: (String) -> String = { it.toUpperCase() }
println(shout("hello"))
```

Features - Collections and Functional Operations

```
val names = listOf("Alice", "Bob", "Charlie")
names.filter { it.startsWith("A") }.forEach { println(it) }
```

Features - when Expression

```
fun describeLength(name: String) = when (name.length) {
    0 -> "Empty name"
    in 1..4 -> "Short name"
    else -> "Long name"
}
println(describeLength("Anna"))
```

Features - Sealed Classes

```
sealed class Response
data class Success(val data: String) : Response()
data class Error(val message: String) : Response()
val response = Success("Data loaded")
when (response) {
    is Success -> println(response.data)
    is Error -> println(response.message)
}
```

Features - Object Declarations (Singletons)

```
object Config { val baseUrl =
"https://api.example.com" }
println(Config.baseUrl)
```

Features - Smart Casts

```
fun printLength(obj: Any) {
    if (obj is String) {
        println(obj.length) // obj is automatically cast to String
    }
}
```

Features - Destructuring Declarations

```
val (firstName, lastName) = person
println(firstName)
```

Agenda

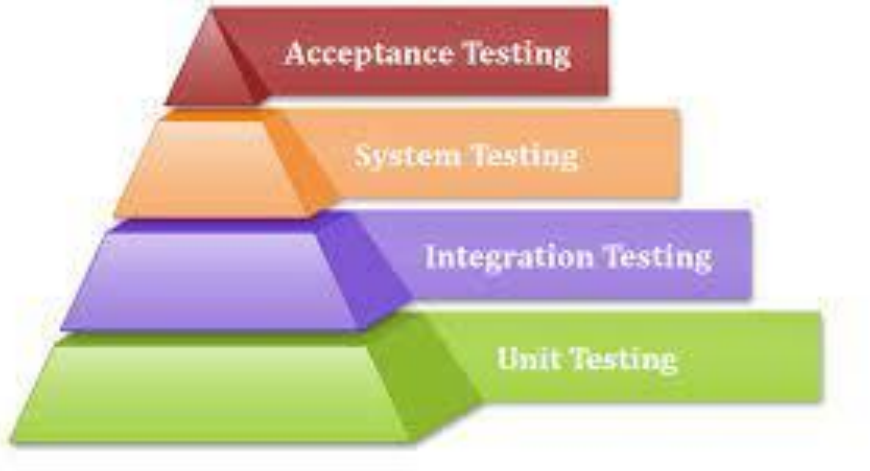
- Kotlin@Kohls
- “Why Kotlin?”
- Features - Demo
- **Adoption Suggestions**
- Open Discussion

Adoption Suggestions

- **Testing**
- POJOs
- Next steps

Adoption Suggestions - Testing

- Introduce Kotlin through unit tests
- Improve conciseness & readability
- Learn idioms w/o risking production



Adoption Suggestions - Unit Testing (JUnit)

```
class CalculatorTest {
    @Test
    fun addingTwoNumbersTogetherShouldBeTheSum() {
        assertEquals(2, Calculator().add(1, 1))
    }
}
class Calculator {
    fun add(x : Int, y : Int) = x + y
}
```

Adoption Suggestions - Unit Testing (JUnit)

```
class CalculatorTest {
    @Test
    @DisplayName("Adding two numbers together should be the sum")
    fun addingTwoNumbersTogetherShouldBeTheSum() {
        assertEquals(2, Calculator().add(1, 1))
    }
}
class Calculator {
    fun add(x : Int, y : Int) = x + y
}
```

Adoption Suggestions - Unit Testing (Junit w/Kotlin sugar)

```
class CalculatorTest {
    @Test
    fun `Adding two numbers together should be the sum`() {
        assertEquals(2, Calculator().add(1, 1))
    }
}
class Calculator {
    fun add(x : Int, y : Int) = x + y
}
```

Adoption Suggestions - Unit Testing (Kotest)

```
class CalculatorTest : StringSpec({
    "Adding two numbers together should be the sum" {
        Calculator().add(1, 1) shouldBe 2
    }
})

class Calculator {
    fun add(x: Int, y: Int) = x + y
}
```

Adoption Suggestions

- Testing
- **POJOs**
- Next Steps

Adoption Suggestions - POJOs

- Replace POJOs with Kotlin data classes
- Reduce boilerplate code
- Enhance code readability & maintainability

```
data class ScienceFictionBook(val title : String, val author : String = "Isaac Asimov")
```

Adoption Suggestions

- Testing
- POJOs
- **Next Steps**

Adoption Suggestions - Next Steps

- Utility functions (extensions)
- Scripting
- Prototyping
- Gradual integration

Agenda

- Kotlin@Kohls
- “Why Kotlin?”
- Features - Demo
- Adoption Suggestions
- **Open Discussion**

Open Discussion

CHRISTOPHER.ROBERTS@KOHLS.COM