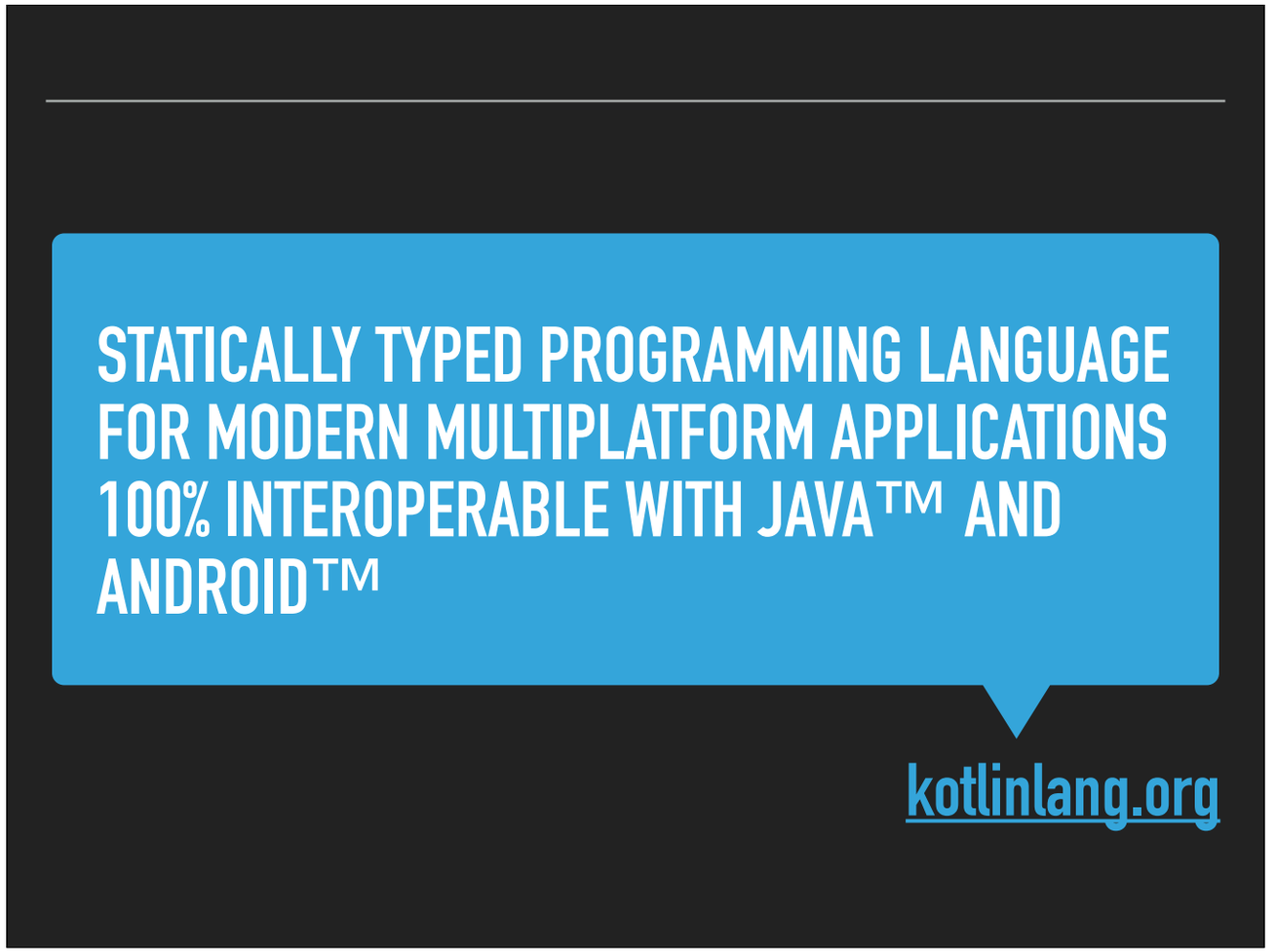
KOTLIN

LEOMINSTER CODE MEETUP

OCT 3 2018

Will Winder
willwinder.com



This is the first thing you see on kotlinlang.org! They could do better.

KOTLIN IS JUST PLAIN FUN. MAYBE IT'S SUBLIMINAL ADVERTISING, SINCE THEIR KEYWORD FOR DECLARING METHODS IS FUN. BUT IT'S SOMEHOW TURNED ME FROM A SURLY PROFESSIONAL PROGRAMMER INTO A HOBBYIST AGAIN.

Steve Yegge

programmer/blogger. Worked at google and amazon

ABOUT KOTLIN

- Developed by Jetbrains, maker of great IDEs
- 100% interoperable with Java
- Started in 2010, open sourced February 2012
- 1.0 released February 15, 2016
- "Kotlin" is an island near St. Petersburg
- First class language for Android in 2017

LANGUAGE GOALS

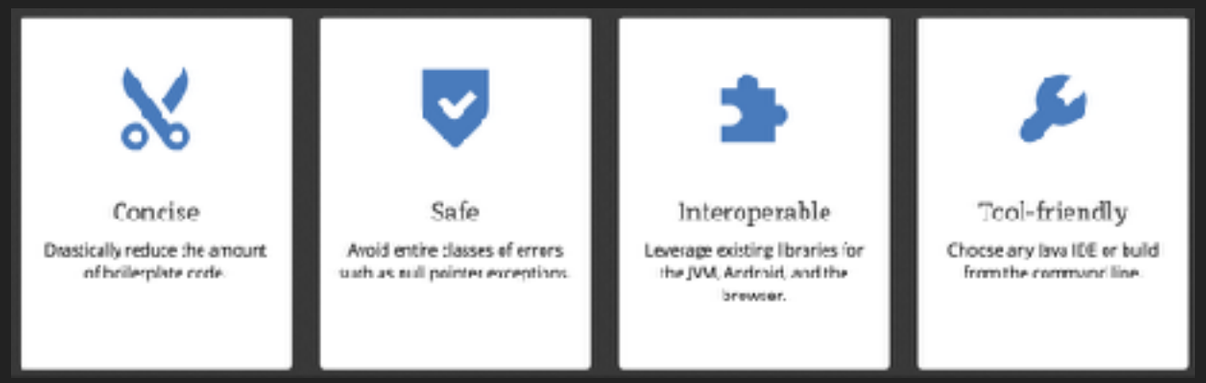
- **General purpose**
- **Statically Typed**
- **Multi-paradigm** - OOP, Functional
- **Multiple Backends** - JVM, Javascript, Native (LLVM)
- **Productivity** - Not a research project
- **Borrow** - Shamelessly from other languages

WHO IS KOTLIN FOR?

- Java developers.
  Generates 100% interoperable byte code, back to Java 6 (2006).
- Web developers.
  Transpile to javascript.
- Windows / iOS developers.
  LLVM backend supports native binaries.
- Developers who write shell scripts!

Pluggable backends currently target
Java bytecode
Javascript
Native
There is also scripting support

WHY KOTLIN



From kotlinlang.org, we'll touch a bit on these

HELLO, WORLD!

```
fun main(args: Array<String>) {
    println("Hello, world!")
}
```

Jumping right in... note that there are no classes here. They are not required with kotlin.

VALUES AND IMMUTABILITY

```
val firstName: String = "Will"

val lastName = "Winder"

println("$firstName $lastName")

// values are immutable!
firstName = "Owen"
```

Values are immutable

IDE Hint:
String type definition is redundant
firstName cannot be reassigned

VARIABLES

```
var firstName = "Will"
var lastName = "Winder"
firstName = "Owen"
println("$firstName $lastName")
```

IDE Hints:
"Will" - redundant initializer
var - can be val, lastName is never modified

String interpolation

NULL SAFETY

```
var x: Int = null
x --

var y: Int? = null
y --
```

Null safety is the killer feature in kotlin

NULL SAFETY

```
// 'String' cannot be null
val nullString1: String = null

// Safe, 'String' cannot be null
nullString1.reversed()

// 'String?' may be null!
val nullString2: String? = null

// Not safe, cannot call a function on a nullable object!
nullString2.reversed()
```

This is important, so here are some more examples

WORKING WITH NULL

```
// Smart casting, the compiler upgrades the
// type to 'String' after the null check.
if (nullString2 != null) nullString2.reversed()

// The compiler also suggests we use '?.' which
// has the same effect.
nullString2?.reversed()?.toUpperCase()
```

Compiler knows when a type has been validated.

"?." operator

WORKING WITH NULL

```
// Illegal assignment, 'result1' cannot be null.
val result1: String = nullString2?.reversed()

// Default assignment in case of a null!
val result2: String = nullString2?.reversed() ?: "default!"

// Or error handling!
val result3: String = nullString2?.reversed() ?: throw Exception("malformed!")
```

Default values and error handling

NULLABLE PARAMETERS

```
fun someFunction(param1: String?, param2: String): Int {
    println(param1.reversed())
    println(param2.padStart(5, '0'))
    val x: Int? = null
    val y: Int = null

    return null
}
```

IDE Hints: unused variables, method can be private
Compiler errors:
- param1 is nullable, must check for null
- y is not nullable
- return value is not nullable

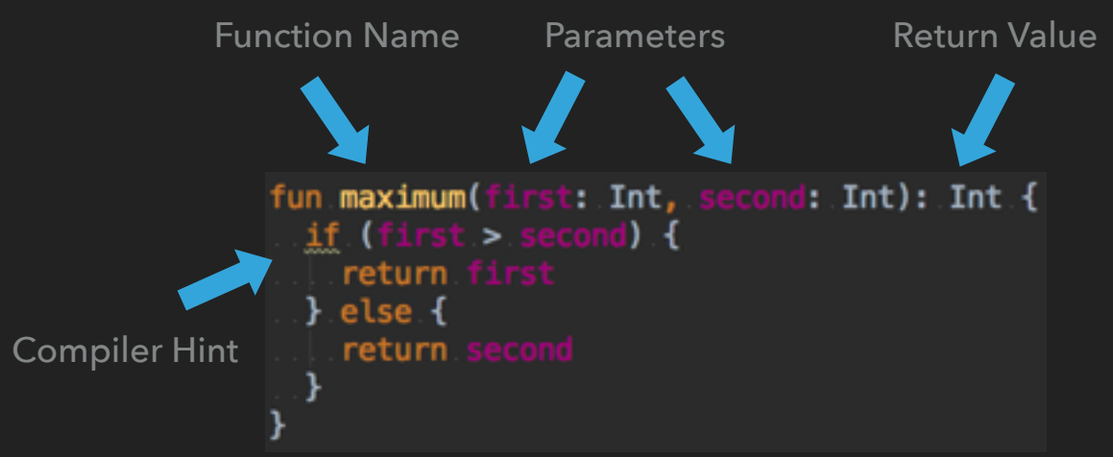
leftPad is built into the standard library

FUNCTION SYNTAX

```
fun maximum(first: Int, second: Int): Int {
    if (first > second) {
        return first
    } else {
        return second
    }
}
```

Explain function syntax briefly, advance to show labels.

FUNCTION SYNTAX



Notice the compiler has some input about that if

IF IS AN EXPRESSION

```
fun maximum(first: Int, second: Int): Int {
  if (first > second) {
    return first
  } else {
    return second
  }
}
```

```
fun maximum(first: Int, second: Int): Int {
  return if (first > second) {
    first
  } else {
    second
  }
}
```

if is an expression, so the IDE Suggest lifting out the return

SINGLE EXPRESSION FUNCTION

```
fun maximum(first: Int, second: Int): Int {
    return if (first > second) {
        first
    } else {
        second
    }
}
```

```
fun maximum(first: Int, second: Int): Int =
    if (first > second) {
        first
    } else {
        second
    }
```

The entire body of this function is a single expression
we can use = and skip the return

SINGLE EXPRESSION FUNCTION

```
fun maximum(first: Int, second: Int): Int =
    if (first > second) {
        first
    } else {
        second
    }
```

```
fun maximum(first: Int, second: Int) =
    if (first > second) first else second
```

The return type can be inferred by the compiler, so that is optional as well.

We can also collapse the if/else to a single line for this simple function

CALLING A FUNCTION

```
fun maximum(first: Int, second: Int) =
    if (first > second) first else second
```

```
fun main(args: Array<String>) {
    println(maximum(1,2))
    println(maximum(first = 1, second = 2))
}
```

Named Parameters

Named parameters.

DATA STRUCTURES: LIST

```
val list: List<Int> = listOf(1, 2, 3, 4, 5)

list.filter { it >= 3 }
    .filterNot { it % 2 == 0 }
    .forEach { println("odd numbers >= 3: $it") }
```

```
odd numbers >= 3: 3
odd numbers >= 3: 5
```

Immutable
Operations are applied to the collection as a stream

DATA STRUCTURES: MUTABLE LIST

```
val mutableList: MutableList<Int> = mutableListOf(0)

(10 downTo 1).forEach { mutableList.add(it) }

(1..5).forEach { index ->
    mutableList.shuffle()
    println("$index. even value: " + mutableList.find { it % 2 == 0 })
}
```

```
1. even value: 0
2. even value: 8
3. even value: 6
4. even value: 8
5. even value: 10
```

Mutable list
Ranges (10 downTo 1), (1..5)

DATA STRUCTURES: MAP

```
val map: Map<Int, String> = mapOf(0 to "Zero", 1 to "One")

println("Map value at 1: ${map[1]}")

println("Undefined map value: ${map[2]}")

println("Default map value 1: ${map.getOrDefault(2, "boo")}")
println("Default map value 2: ${map[2] ?: "boo"}")

val mutableMap = mutableMapOf<Int,String>()

mutableMap[0] = "Zero"
```

```
Map value at 1: One
Undefined map value: null
Default map value 1: boo
Default map value 2: boo
```

Map have a mutable variant as well.

DATA STRUCTURES: ARRAY

```
val array: Array<Long> = arrayOf(0, 1, 2)
println(array
    .map { it * it }
    .joinToString())
```

```
0, 1, 4
```

Arrays work pretty much the same as a list.

DATA STRUCTURES: SET

```
val set: Set<String> = setOf("one", "one", "two")

println("Size of set: ${set.size}")
```

```
Size of set: 2
```

This is getting repetitive...

DEFAULT ARGUMENTS

```
fun main(args: Array<String>) {
  /*
  fun <T> Iterable<T>.joinToString(
    separator: CharSequence = ", ",
    prefix: CharSequence = "",
    postfix: CharSequence = "",
    limit: Int = -1,
    truncated: CharSequence = "...",
    transform: (T) -> CharSequence = null
  ): String
   */

  val data = listOf(1, 2, 3, 4)
  println(data.joinToString(", ") { "num($it)" })
  println(data.joinToString(
      separator = " ",
      limit = 2,
      truncated = "and many more"))
}
```

```
num(1), num(2), num(3), num(4)
1 2 and many more
```

Built-in helper function (actually an extension method!)
Last argument is a lambda (common pattern)

CLASSES

```
class MyClass(val text: String, private val count: Int) {
    fun repeat() =
            (1..count).fold(StringBuilder()) {
                acc, _ -> acc.append(text)
            }.toString()
}

fun main(args: Array<String>) {
    val myClass = MyClass("Test", 5)
    println("text: " + myClass.text)
    // count is private
    // println("count: " + myClass.count)
    println(myClass.repeat())
}
```

```
text: Test
TestTestTestTestTest
```

Small class with 2 member variables and a function.

BRINGING IT ALL TOGETHER 1

```
fun mightBeNull(str: String): String? =
    if (Random().nextInt() % 2 == 0) str else null

fun dealingWithNull(input: String) {
    (0..10).forEach {
        println(mightBeNull(input)?.toUpperCase() ?: "it was null")
    }
}

fun main(args: Array<String>) {
    dealingWithNull("Leominster Code Meetup")
}
```

```
LEOMINSTER CODE MEETUP
it was null
LEOMINSTER CODE MEETUP
it was null
it was null
LEOMINSTER CODE MEETUP
LEOMINSTER CODE MEETUP
it was null
LEOMINSTER CODE MEETUP
LEOMINSTER CODE MEETUP
LEOMINSTER CODE MEETUP
```

IDE Hints: dealingWithNull / mightBeNull can be private
Range: (0..10)
?. and ?: Elvis operator

BRINGING IT TOGETHER 2

```
fun main(args: Array<String>) {
    // Helper function
    fun illegalNumber(num: Int) = num == 3

    // Data class
    data class NumTuple(val num: Int, val name: String? = null)

    val one = NumTuple(1, "one")
    val two = NumTuple(2)

    val listOfLists: List<List<NumTuple>> =
        listOf(
            listOf(two, NumTuple(4, "four")),
            listOf(one, NumTuple(3, "three"))
        )

    listOfLists
        .flatMap { it }
        .sortedBy { it.num }
        .filterNot { illegalNumber(it.num) }
        .forEach { println(it.name ?: it.num) }
}
```

```
one
2
four
```

Data class
Private helper function
List
Streaming processing

JAVASCRIPT

```
<!DOCTYPE html>
<html lang="en">
<body>

<input type="text" name="email" id="email"/>

<!-- kotlin artifacts -->
<script type="text/javascript" src="out/produ
<script type="text/javascript" src="out/produ
</body>
</html>
```

```
import org.w3c.dom.HTMLInputElement
import kotlin.browser.document

fun main(args: Array<String>) {
    document.body?.style?.backgroundColor = "powderblue"
    val email = document.getElementById("email") as HTMLInputElement
    email.value = "wwinder.unh@gmail.com"
}
```

wwinder.unh@gmail.com

Example of javascript manipulating the DOM

WEBSERVER CREATE/READ (JAVALIN)

```
/opt/JavalinTest$ curl -s -X GET http://localhost:7000/user/2 | json_pp
{
   "id" : 2,
   "name" : "corey",
   "sign" : "Virgo"
}
/opt/JavalinTest$ curl -X PUT -d '{"id":3,"name":"ben","sign":"Aquarius"}' \
> http://localhost:7000/user/create -w "\n"
User created successfully: User(id=3, name=ben, sign=Aquarius)
/opt/JavalinTest$
/opt/JavalinTest$ curl -s -X GET http://localhost:7000/user/3 | json_pp
{
   "sign" : "Aquarius",
   "name" : "ben",
   "id" : 3
}
```

Webserver with Create / Read endpoints

```
import io.javalin.Javalin

data class User(val id: Int, val name: String, val sign: String)

fun main(args: Array<String>) {
    val data = mutableMapOf(
        1 to User(1, "will", "Aries"),
        2 to User(2, "corey", "Virgo")
    )

    // Create a server and start it.
    val app = Javalin.create().start(7000)

    // Endpoint handlers
    app.get("/") { ctx -> ctx.result("Hello World") }

    app.get("/user/:id") { ctx ->
        val id = ctx.pathParam("id").toInt()
        data[id]?.let {
            ctx.json(it)
        } ?: ctx.json("not found")
    }

    app.put("/user/create") { ctx ->
        val user = ctx.validatedBody<User>()
            .check({ !data.containsKey(it.id) }, errorMessage = "This user id already exists.")
            .getOrThrow()
        data[user.id] = user
        ctx.status(201).result("User created successfully: $user")
    }
}
```

Just a main function
Data class + automatic JSON Serialization/Deserialization
Create validation!

NOT COVERED...

- **Generics** - Improvements over java, like "reified" types.
- **Coroutines** - Concurrency inspired by C#, Javacript, Go, ...
- **Debugging** - Breakpoints, Profiler, Chrome extension.
- **Exceptions** - Simplified handling for unusual errors.
- **Builders** - For creating your own DSL libraries.
- **Tools** - Monitoring, Testing, Static Analysis, CI, Gradle/Maven, ...
- **Reflection** - Runtime class analysis, enables easy to use libraries.
- **Java** - Tons of syntactic sugar improving java usability.

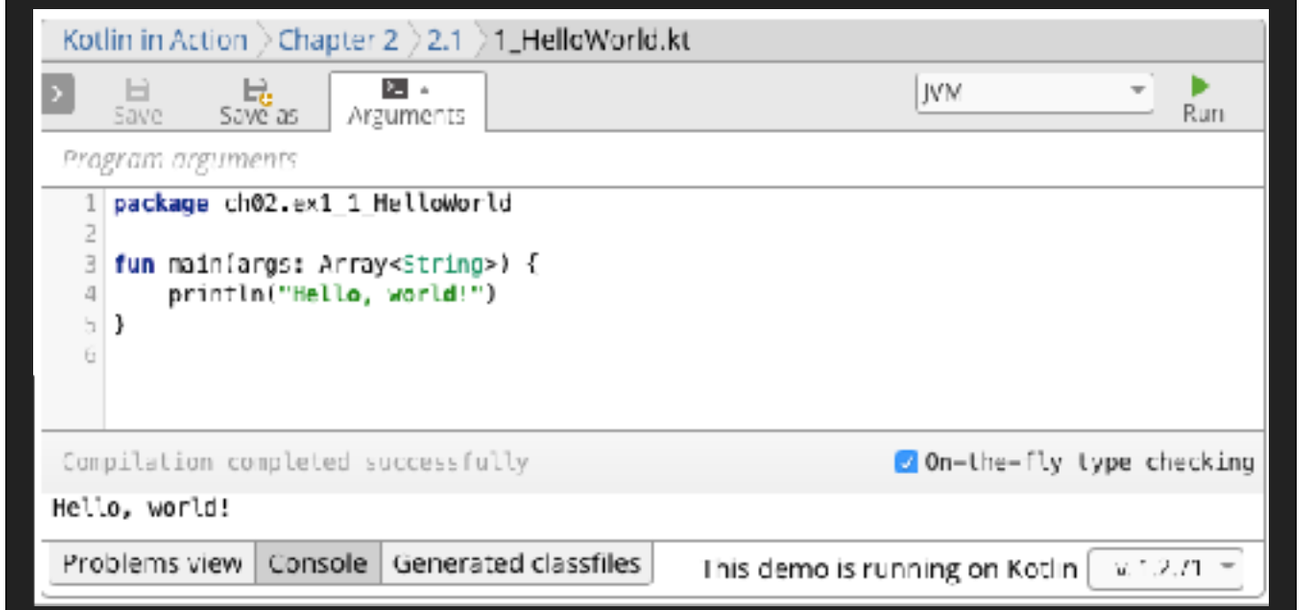
Every other Kotlin presentation is going to start with the Java improvements :)

HOW TO LEARN

- **Books** - https://kotlinlang.org/docs/books.html
- **Kotlin Koans** - "Hands-on" exercise approach.
- **Try Kotlin** - Same web app as Kotlin Koans, self driven.
- **KScript** - Ease into it with small scripts
- **Documentation** - https://kotlinlang.org/docs/reference/
- **For Java Devs** - IntelliJ's "Convert Java File to Kotlin File"

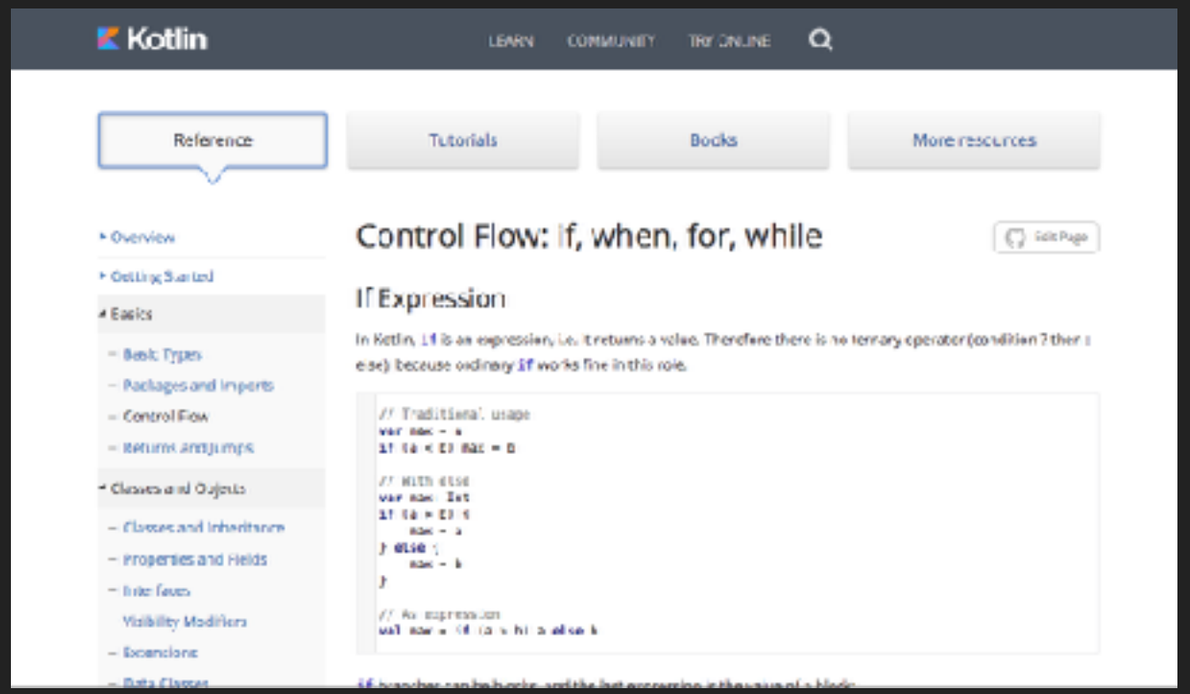
Lots of great resources

HOW TO LEARN – WEB APP



Webapp is a little buggy, but has lots of examples, and a "Koan" section with quizzes

HOW TO LEARN – DOCUMENTATION



Reference documentation is pretty good.

HOW TO LEARN – DOCUMENTATION

For Loops

`for` loop iterates through anything that provides an iterator. This is equivalent to the `foreach` loop in languages like C#. The syntax is as follows:

```
for (iter in collection) print(item)
```

The body can be a block.

```
for (iter: Int in ints) {
    // ...
}
```

As mentioned before, `for` iterates through anything that provides an iterator, i.e.

- has a member- or extension-function `iterator()`, whose return type
  - has a member- or extension-function `next()`, and
  - has a member- or extension-function `hasNext()` that returns `Boolean`.

Even contains runnable / editable inline snippets

HOW TO LEARN - KSCRIPT

```
1 #!/usr/bin/env kotlinc -script
2
3 import java.io.File
4
5 val folders: Array<out File> =
6     File(args[0]).listFiles { file -> file.isDirectory }!!
7
8 folders.forEach { folder -> println(folder) }
```

```
/opt/KotlinJavascriptProgram$ ./dir.kts /opt/KotlinJavascriptProgram/
/opt/KotlinJavascriptProgram/.idea
/opt/KotlinJavascriptProgram/out
/opt/KotlinJavascriptProgram/src
```

No need to worry about build scripts this way.
Using an IDE is still recommended though.

LIBRARIES

- Thousands of Java libraries.
- Micro Web Frameworks - Jooby, Javalin, Spark, Ktor
- Testing - Spek, JUnit, TestNG, Mockito, AssertJ
- Documentation - Dokka
- Build Tools - Gradle, Maven
- JSON - Jackson, Gson

LINKS

- Blog post "Why Kotlin Is Better Than Whatever Dumb Language You're Using"
  https://steve-yegge.blogspot.com/2017/05/why-kotlin-is-better-than-whatever-dumb.html
- https://kotlinlang.org/
- Huge list of libraries, websites, tutorials, books, and other resources
  https://github.com/mcxiaoke/awesome-kotlin
- Some inspiration from this presentation:
  https://speakerdeck.com/alexgherschon/introduction-to-kotlin
- IntelliJ IDE, Community Edition is free:
  https://www.jetbrains.com/idea
- My webpage: https://willwinder.com
- My github: https://github.com/winder