Facts you may not know About Kotlin ,)

Eugene Petrenko
eugene.petrenko@jetbrains.com
@jonnyzzz

I’m **Eugene Petrenko**

- or just **@jonnyzzz**
- I’ve been using Kotlin when
  - `trait` keyword was used for interfaces
  - `init` keyword was not yet invented

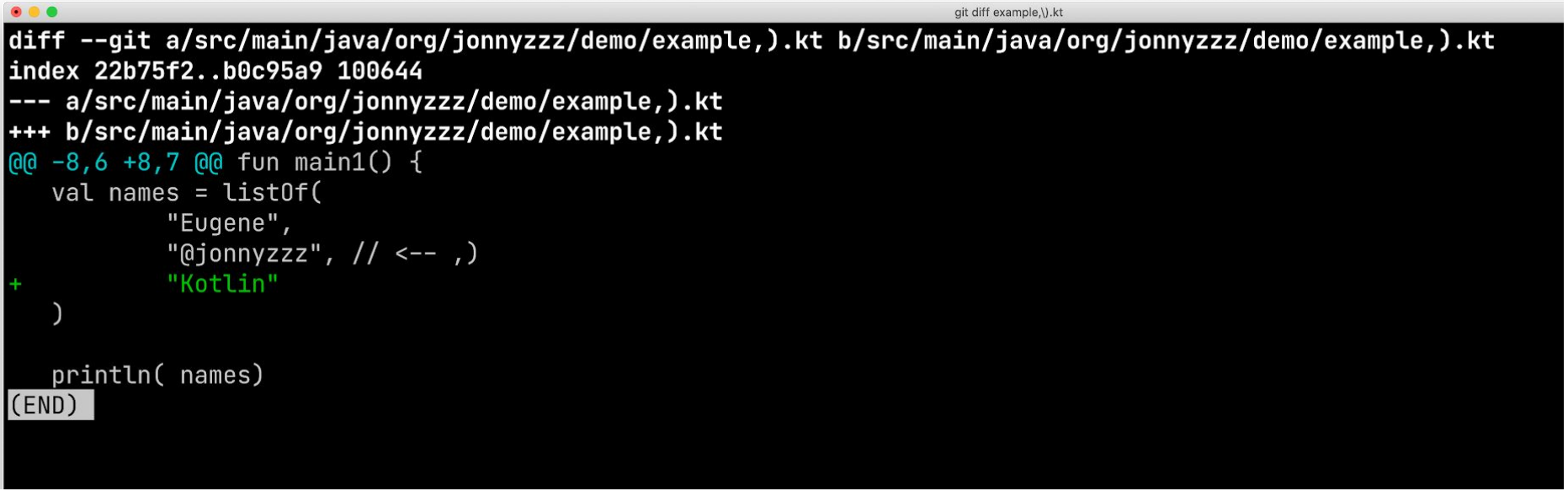
Kotlin 1.4 ,)

```
val names = listOf(
        "Eugene",
        "@jonnyzzz", // <-- ,)
)
```

Kotlin 1.4 ,)

```
val names = listOf(
        "Eugene",
        "@jonnyzzz", // <--- no line change to
        "Kotlin"
)
```

Kotlin 1.4 ,)


git diff example,\).kt
diff --git a/src/main/java/org/jonnyzzz/demo/example,).kt b/src/main/java/org/jonnyzzz/demo/example,).kt
index 22b75f2..b0c95a9 100644
--- a/src/main/java/org/jonnyzzz/demo/example,).kt
+++ b/src/main/java/org/jonnyzzz/demo/example,).kt
@@ -8,6 +8,7 @@ fun main1() {
val names = listOf(
"Eugene",
"@jonnyzzz", // <-- ,)
+ "Kotlin"
)
println(names)
(END)

Why Kotlin?

Why Kotlin?

Concise

Drastically reduce the amount of boilerplate code.

See example

Safe

Avoid entire classes of errors such as null pointer exceptions.

See example

Interoperable

Leverage existing libraries for JVM, Android and the browser.

See example

Tool-friendly

Choose any Java IDE or build from the command line.

See example

JetLang or Jet

- The code name
  - “jet” in config files under .idea
- **Jet** prefix for many internal projects at JetBrains
- `*.jet` files extension

Code Style Settings under `.idea`

```
<codeStyleSettings language="jet">
    <indentOptions>
        <option name="INDENT_SIZE" value="2"/>
        <option name="TAB_SIZE" value="2"/>
    </indentOptions>
</codeStyleSettings>
```

Codename Kotlin

- Public announcement
  - JVMLS 2011
- `*.kt` file extension
  - translations are not good for `*.kot`

Google Maps https://www.google.com/maps/place/Kotlin/@51.9147882,17.648726,3a,75y,90t/data=!3m8!1e2!3m6!1sAF1QipOZfLiZdd62EIGQ5biWh-y3lICadzACvotpAPXR!2e10!3e12!6shttps:%2F%2Flh5.googleusercontent.com%2Fp%2FAF

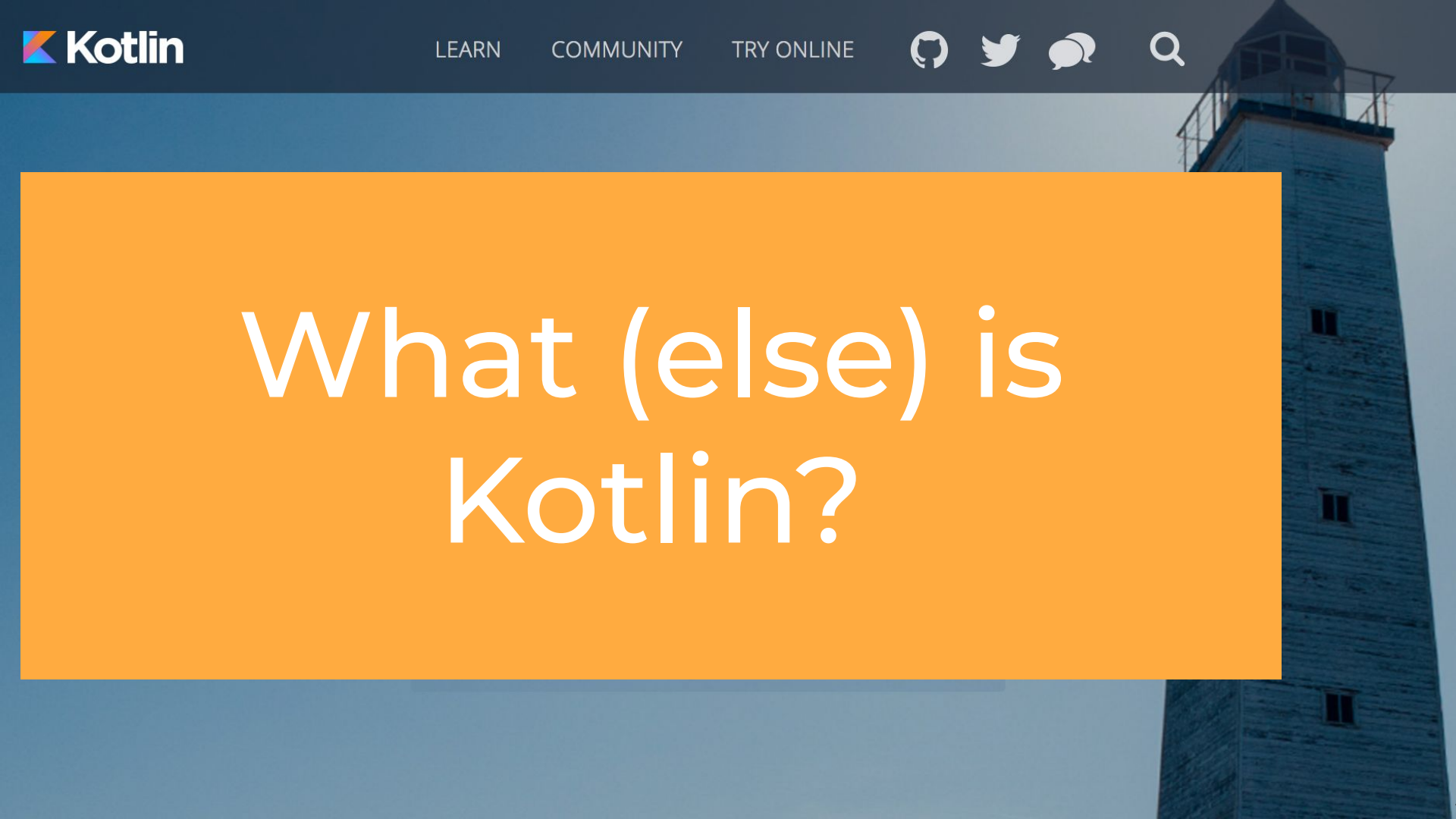
kotlinlang.org

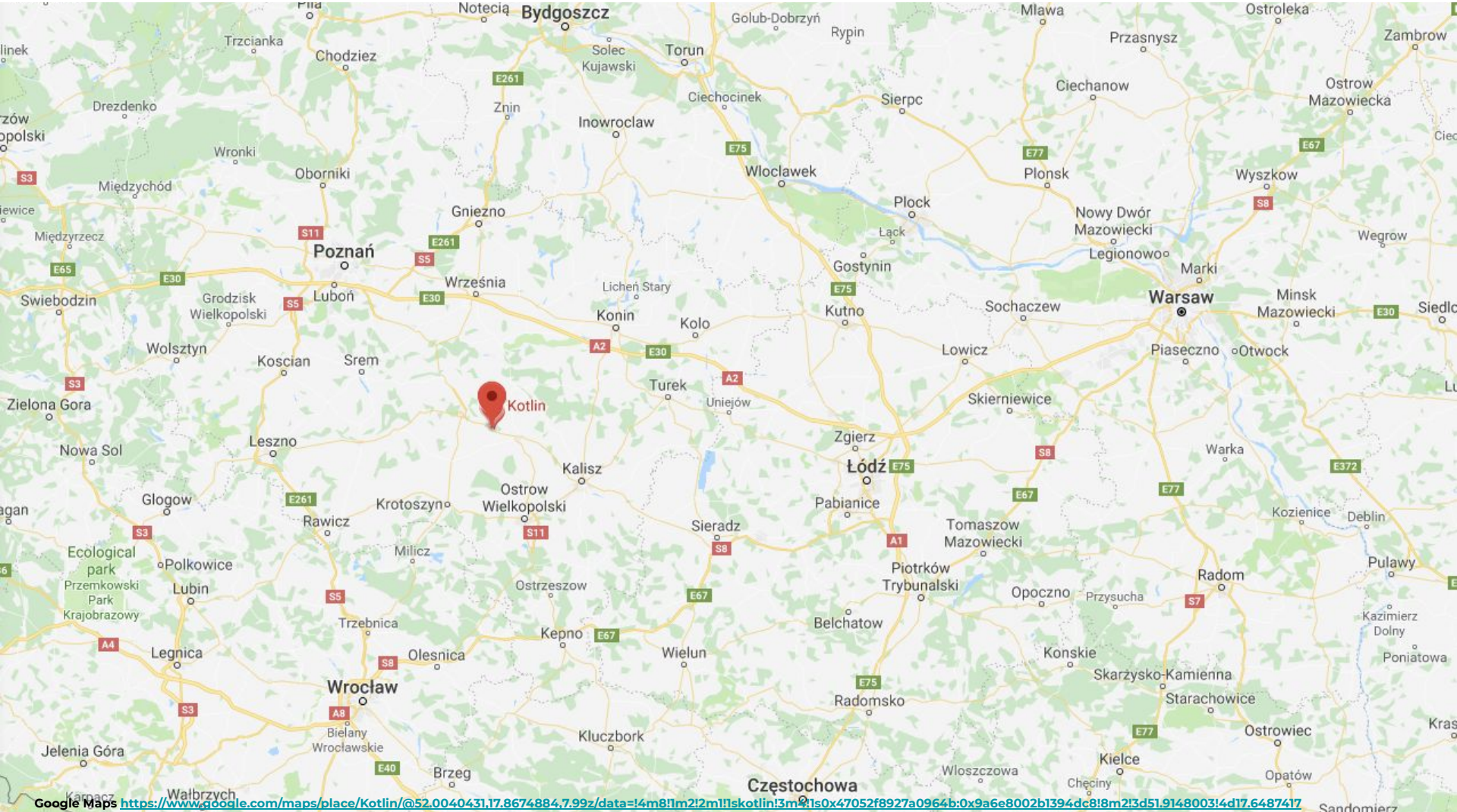
since 2014


PROJECT
Kotlin
Docs
Contribute
Try Online
Statically typed programming language
targeting the JVM and JavaScript
100% interoperable with Java™
TRY KOTLIN

Kotlin 1.0

since February, 2016


Kotlin
LEARN
COMMUNITY
TRY ONLINE
What (else) is Kotlin?



Google Maps https://www.google.com/maps/place/Kotlin/@52.0040431,17.8674884,7.99z/data=!4m8!1m2!2m1!1skotlin!3m4!1s0x47052f8927a0964b:0x9a6e8002b1394dc8!8m2!3d51.9148003!4d17.6487417

Kotlin v1.4.0

Learn Community Teach Play ↗

A modern programming language that makes developers happier. Open source forever

Get started Try online

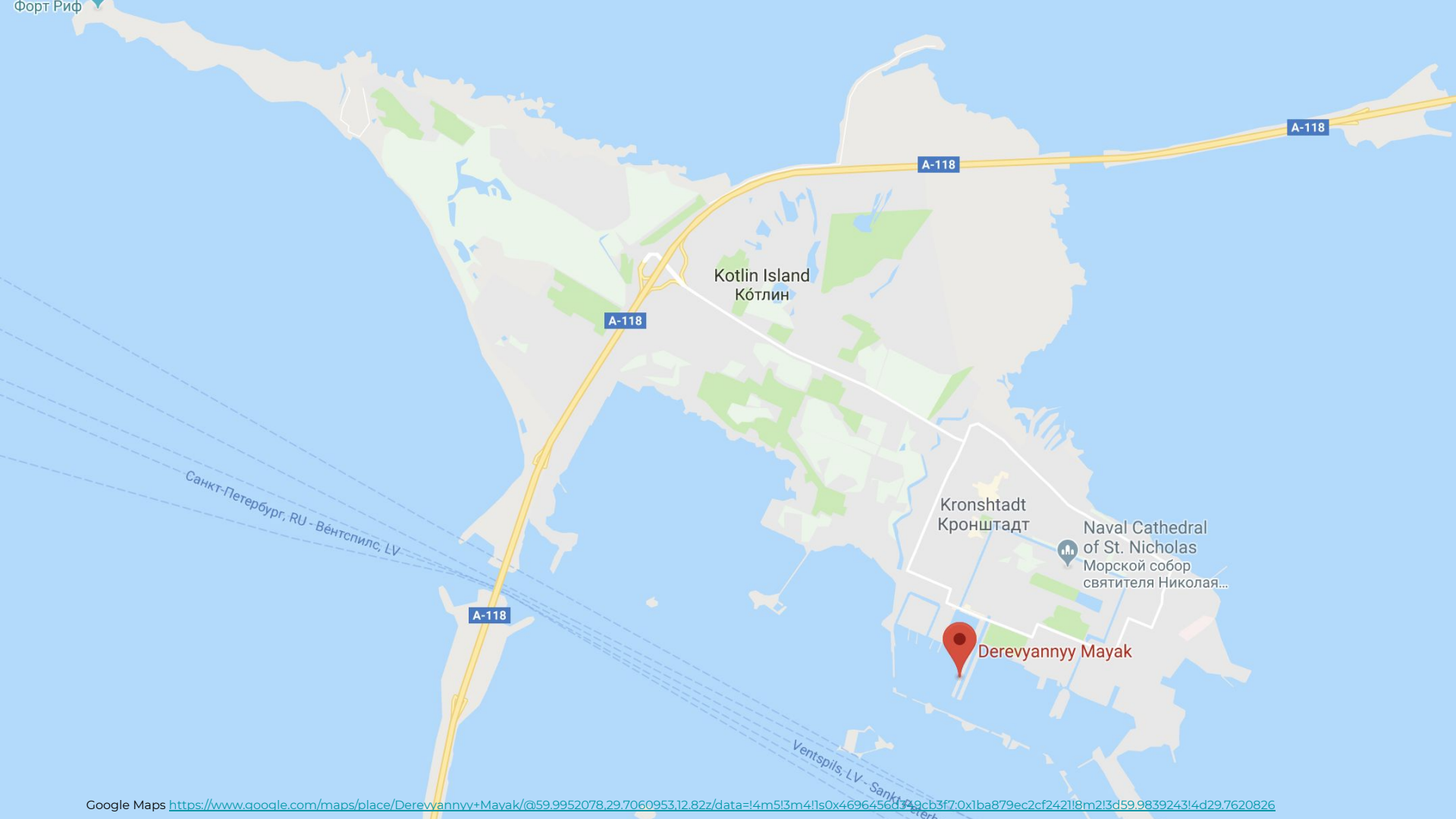
Good for

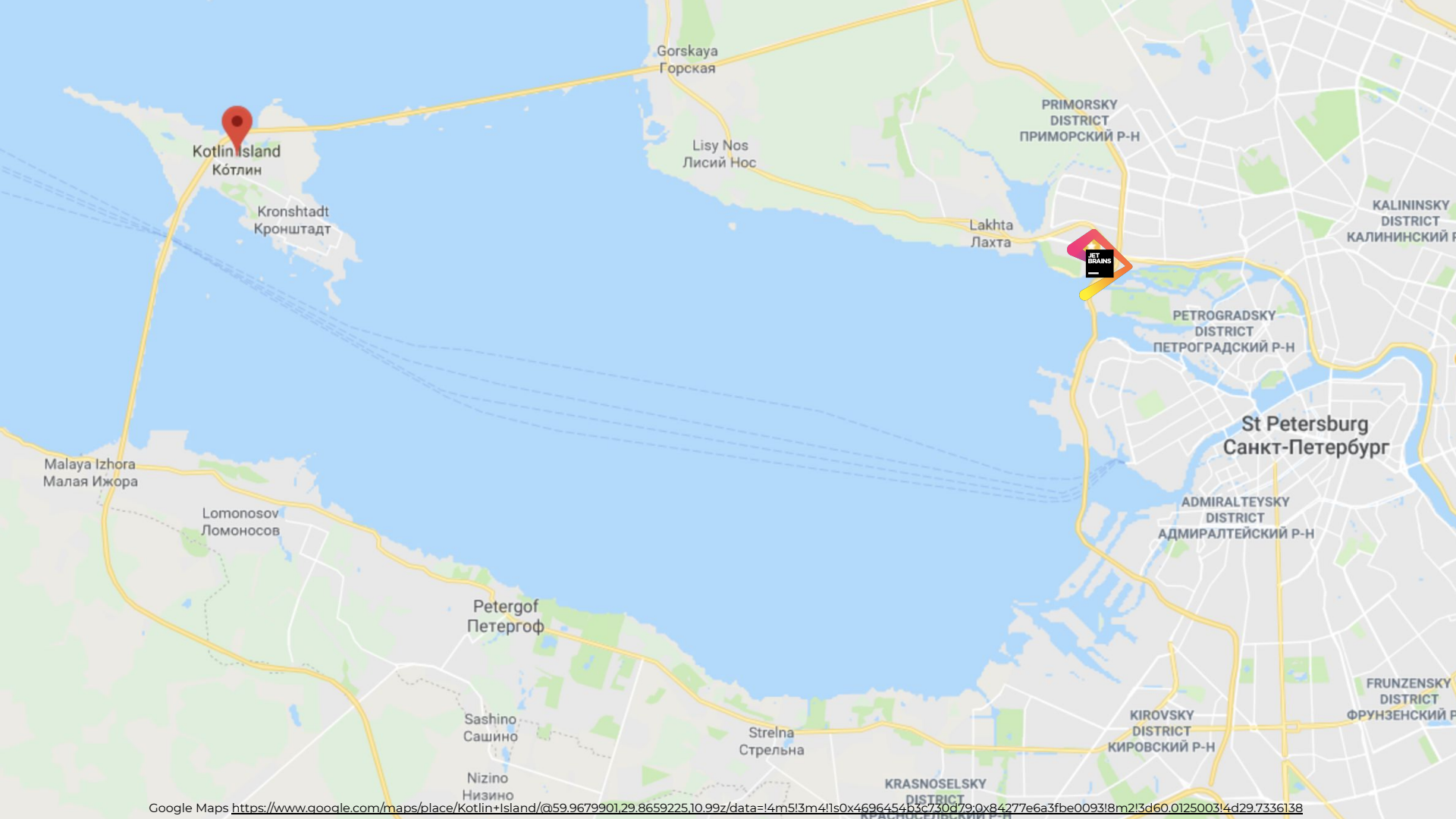
- Mobile cross-platform →
- Native →
- Data science →
- Server-side →
- Web development →
- Android →


Kotlin
LEARN
COMMUNITY
TRY ONLINE
Statically typed programming language for modern multiplatform applications
100% interoperable with Java™ and Android™
Now official on Android

Kotlin Island outside of Saint Petersburg, Russia (SOURCE: Wikimedia)

Дарья Орешкина, Спутник http://turizm.sputnik.ru/items/derevyanniy-mayak-v-kronshtadte

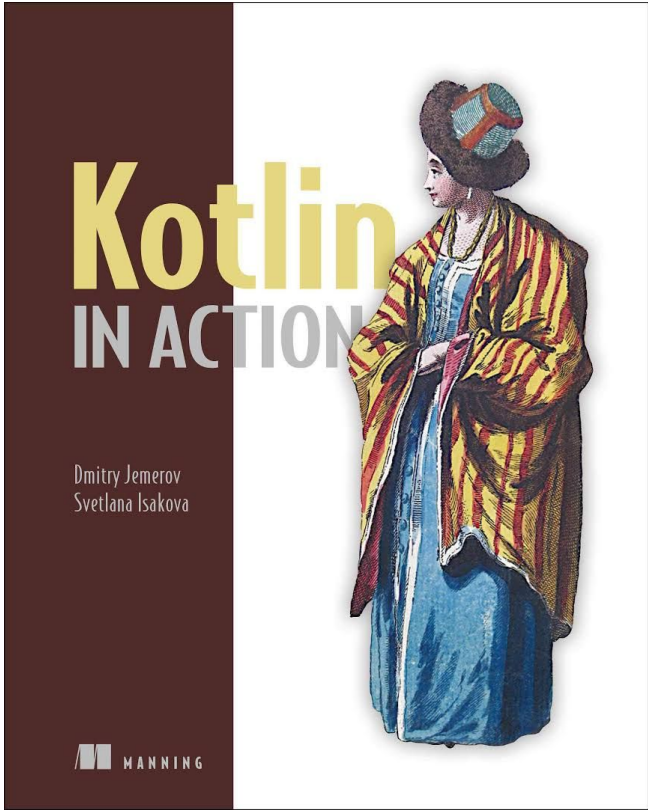




Google Maps https://www.google.com/maps/place/Kotlin+Island/@59.9679901,29.8659225,10.99z/data=!4m5!3m4!1s0x469645463c730d79:0x84277e6a3fbe0093!8m2!3d60.0125003!4d29.7336138

… Kotlin is **named after an island** near St. Petersburg, Russia, where most of the Kotlin development team is located

… We followed the precedent established by Java and Ceylon, but we decided to go for something closer to our homes



Kotlin Island (Kronstadt city)

- near Saint Petersburg, Russia
- 32 km away from JetBrains office

Warm-up

```
val x = 0 + 0
println(x)
```

play.kotlinlang.org

```
val x = null + null

// what will it print?
println(x::class)
println(x)
```

```
/**
 * Concatenates this string with the
 * string representation of the given
 * [other] object. If either the receiver
 * or the [other] object are null, they are
 * represented as the string "null".
 */
operator fun String?.plus(other: Any?): String
```

Overloaded Operator `plus`

```
val x = null.plus(null)
```

```
// what will it print?
println(x::class)
println(x)
```

“null” for null

- It is easier to see a `null` value
- We got used to that behaviour
- `Hello null,` (have you seen that?)
- A puzzler

“nullnull”

The answer is …

```
val x = null + null

// class kotlin.String
println(x::class)
// nullnull
println(x)
```

```
operator fun Nothing?.plus(x: Any?): Nothing
```

```
operator fun Nothing?.plus(x: Any?): Nothing
```

```
val x = null + null
```

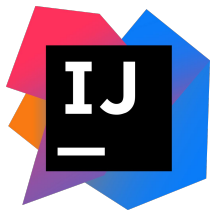
[UNREACHABLE_CODE] Unreachable code

[UNUSED_VARIABLE] Variable 'x' is never used

Remove variable 'x' ⌥⇧⏎ More actions... ⌥⏎

val x: Nothing

```
// W                                    nt?
println(x::class)
println(x)
```



```
@Deprecated
operator fun
```

```
@Deprecated( message: "fix me", level = ERROR)
operator fun Nothing?.plus(x: Any?): Nothing
```

```
val x = null + null
```

[DEPRECATION_ERROR] Using 'plus(Any?): Nothing' is an error. fix me

```
// what will it print?
println(x::class)
println(x)
```

Nothing

Nothing type in Kotlin

- The type has **no values**
- Subtype of any other type
- Means execution is terminated
  - e.g. exception is thrown

```
fun ourSpecialHardStopFunction(): Nothing {
    throw Exception("assert failed")
}

fun example() {
    ourSpecialHardStopFunction()
    println("It will not run")
}
```

[UNREACHABLE_CODE] Unreachable code

Nothing Expressions

```
val a = return "@jonnyzzz"
val b = throw Exception("...")
val c = TODO( reason: "Not ready yet")
val d = ourSpecialHardStopFunction()
val e = exitProcess( status: 1)

/// a,b,c,d,e type is Nothing
```

```kotlin
/**
 * Always throws [NotImplementedError] stating that
 * operation is not implemented.
 */
fun TODO(reason: String): Nothing {
  throw NotImplementedError("An operation is not implemented: $reason")
}
```

TODO(): Nothing as Impl

```
fun findItem(id: Id): Item? = TODO("$id")

fun Item.loadInfo(): String? = TODO()
```

```
fun loadMoreInfo(id: Id): String? {
  val item = findItem(id)
  if (item == null) return null

  val info = item.loadInfo()
  if (info == null) {
    throw Exception("No Info for $id")
  }
  return info
}
```

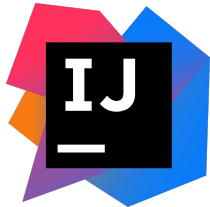
Replace 'if' with elvis operator more... (⌘F1)

```
fun loadMoreInfo(id: Id): String? {
    val item = findItem(id)
    if (item == null) return null

    val info = item.loadInfo()
    if (info == null) {
        o for $id")
    }
    return info
}
```

Replace 'if' with elvis operator more... (⌘F1)

Replace 'if' with elvis operator

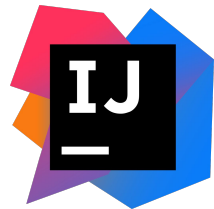
Add braces to 'if' statement



```
fun loadMoreInfo(id: Id): String? {
    val item = findItem(id)
    if (item == null) return null

    val info = item.loadInfo() ?: throw Exception("…")
    return info
}
```

Replace 'if' with elvis operator ▶

Add braces to 'if' statement ▶



```
fun loadMoreInfo(id: Id): String? {
  val item = findItem(id) ?: return null
  return item.loadInfo() ?: throw Exception("…")
}
```

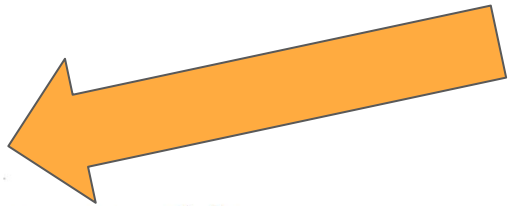
```
fun loadMoreInfo(id: Id): String? {
  val item = findItem(id) ?: return null
  return item.loadInfo() ?: throw Exception("…")
}
```

```
fun loadMoreInfo(id: Id): String? {
  val item = findItem(id)
  if (item == null) return null

  val info = item.loadInfo()
  if (info == null) {
    throw Exception("No Info for $id")
  }
  return info
```

Replace 'if' with elvis operator more... (⌘F1)

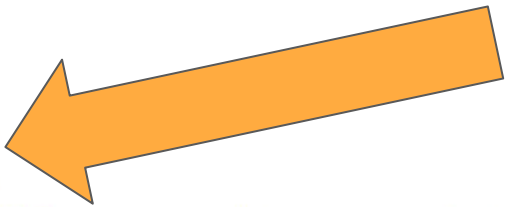
Shorter (but different) - ?.

```
fun loadMoreInfo(id: Id) =
        findItem(id)?.loadInfo()
```



Functional with `let`

```
fun loadMoreInfo(id: Id) =
        findItem(id)?.let {
          it.loadInfo() ?: throw Exception("…")
        }
```



Nothing (type) helps to code clean

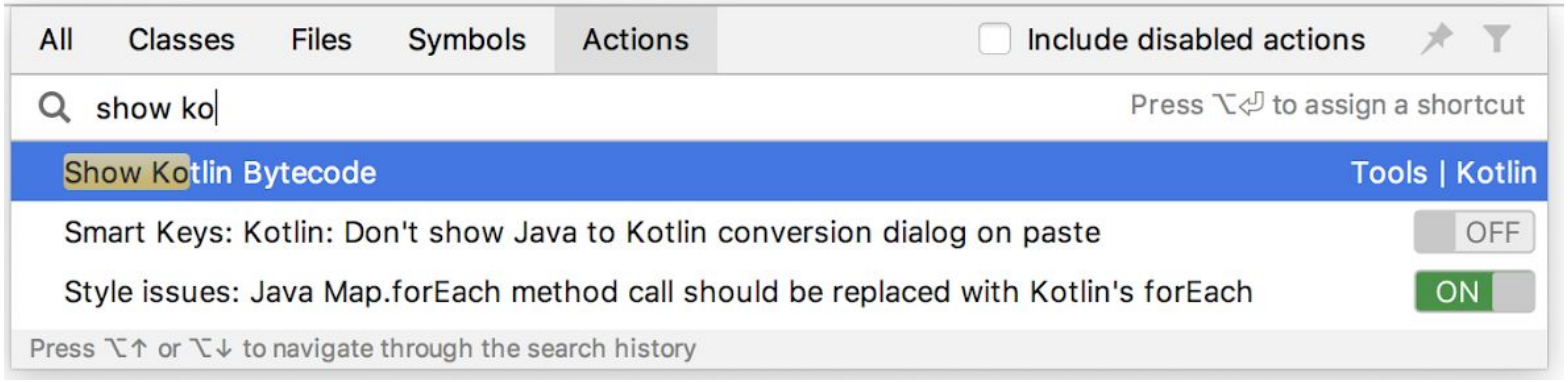
How do Generics Work?

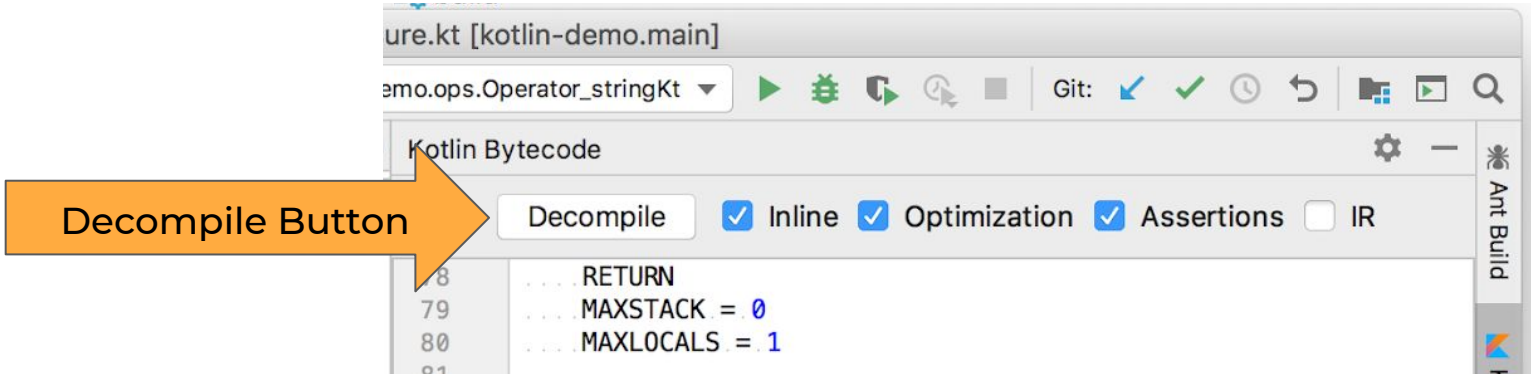
Consider a *generic* fun

```
fun <T> genericFunction(t: T): List<String>
        where T : Number,
              T : Runnable {
  t.run()
  return listOf("123" + t.toDouble())
}
```

Show Kotlin Bytecode

CMD/CTRL + SHIFT + A





Decompile Button

Decompiled as Java

```java
@NotNull
public static final List genericFunction(@NotNull Number t) {
    Intrinsics.checkParameterIsNotNull(t, "t");
    ((Runnable)t).run();
    return CollectionsKt.listOf("123" + t.doubleValue());
}
```

```
fun <T> hackCast(t: Any): T {
    return t as T
}

inline fun <reified Y> reifiedCast(t: Any): Y {
    return t as Y
}
```

Unchecked cast: Any to T

Will it crash?

```
fun <T> hackCast(t: Any): T = t as T

inline fun <reified Y> reifiedCast(t: Any): Y = t as Y


hackCast<Int>("not int")

reifiedCast<Int>("crash")
```

Decompiled as Java

Decompile

```
public static final Object hackCast(@NotNull Object t) {
    Intrinsics.checkParameterIsNotNull(t, "t");
    return t;
}

static {
    boolean var0 = false;
    boolean var1 = false;
    hackCast("not int");
    Object t$iv = "crash";
    Integer var10000 = (Integer)t$iv;
    test = Unit.INSTANCE;
}
```

cast erased (!)

castToY is inlined!

Type Erasure

- Replace <T> with an upper type
- Insert type casts
- Add bridge methods for substituted generics (for implementing members)

The Erased Workarounds

- Do not cast to `T` (cast is erased)
- Pass `Class<T>` as a parameter
  - call `Class<T>#cast()` method

Reified Generics

- inline function
  - method is inlined to the call site
  - lambdas are inlined too
- `<reified T>` are substituted with the actual types

Reading JSON

```
inline fun <reified T> load(text: String): T {...}

fun main() {
  data class MyData(val huge: List<Map<String, List<Int>>>)

  //type inference rocks!
  val clazz = MyData(load("[{\"a\":[42]}]"))
  println(clazz)

  //explicit generic arguments
  println(load<List<String>>("[\"s\"]"))
}
```

History of JVM Generics

Background: how we got the generics we have (Or, how I learned to stop worrying and love erasure). Brian Goetz, June 2020

http://cr.openjdk.java.net/~briangoetz/valhalla/erasure.html

Coroutines with Kotlin

kotlinx.coroutines

- easy to integrate
- supports
  - Futures, Guava, Rx, Android, JDK8, JS,...
- Coroutines, Channels, Select, Flows, etc.

- common — common coroutines across all platforms:
  - `launch` and `async` coroutine builders;
  - `Job` and `Deferred` light-weight future with cancellatio
  - `MainScope` for Android and UI applications.
  - `Dispatchers` object with `Main` dispatcher for Android/ coroutines;
  - `delay` and `yield` top-level suspending functions;
  - `Channel` and `Mutex` communication and synchronizati
  - `coroutineScope` and `supervisorScope` scope builders
  - `SupervisorJob` and `CoroutineExceptionHandler` for s
  - `select` expression support and more.
- core — Kotlin/JVM implementation of common coroutines w
  - `Dispatchers.IO` dispatcher for blocking coroutines;
  - `Executor.asCoroutineDispatcher()` extension, custor
- test — test utilities for coroutines
  - `Dispatchers.setMain` to override `Dispatchers.Main` i
- debug — debug utilities for coroutines.
  - `DebugProbes` API to probe, keep track of, print and dun
- js — Kotlin/JS implementation of common coroutines with `P`
- native — Kotlin/Native implementation of common coroutine
- reactive — modules that provide builders and iteration supp
  - Reactive Streams, RxJava 2.x, and Project Reactor.
- ui — modules that provide coroutine dispatchers for various
  - Android, JavaFX, and Swing.
- integration — modules that provide integration with various
  - JDK8 `CompletableFuture`, Guava `ListenableFuture`,
  - SLF4J MDC integration via `MDCContext`.

Recursion

```
fun sum(x: Long): Long {
    if (x <= 1) return 1
    return 1 + sum(x - 1)
}

fun main() {
    println(sum(100_000))
}
```


?

StackOverflowError

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_171.jdk/Contents/Home/bin/java -Xss160k "-javaagent:/Users/
Exception in thread "main" java.lang.StackOverflowError
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:7)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
	at org.jonnyzzz.demo.corecursionProblem.Co_recursion_problemKt.sum(co-recursion-problem.kt:8)
```

StackOverflow

- check StackOverflow
- add `-Xss1G` for the JVM
- `tailrec` keyword in Kotlin (if possible)
- replace calls with loops
  - move stack to heap manually
- read Roman Elizarov's post

Co-Recursion is OK

```
suspend fun sum(x: Int): Int {
    //free call stack - suspend it
    yield()

    if (x <= 1) return 1
    return 1 + sum(x - 1)
}

fun main() = runBlocking {
    println(sum(100_000))
}
```

Co-Recursion

- `yield()` will free call stack
  - callback added to heap & executed
- `runBlocking` is thread safe
  - may be slower
- easy to code
  - compiler implements stack on heap

Deep Recursive Coroutine (1.4)

```
val deepSum = DeepRecursiveFunction<Long, Long> { x ->
  if (x <= 1) return@DeepRecursiveFunction 1
  1L + callRecursive( value: x - 1) ^DeepRecursiveFunction
}

fun main()  {
  println(deepSum( value: 100_000))
}
```

Learn and Try Coroutines

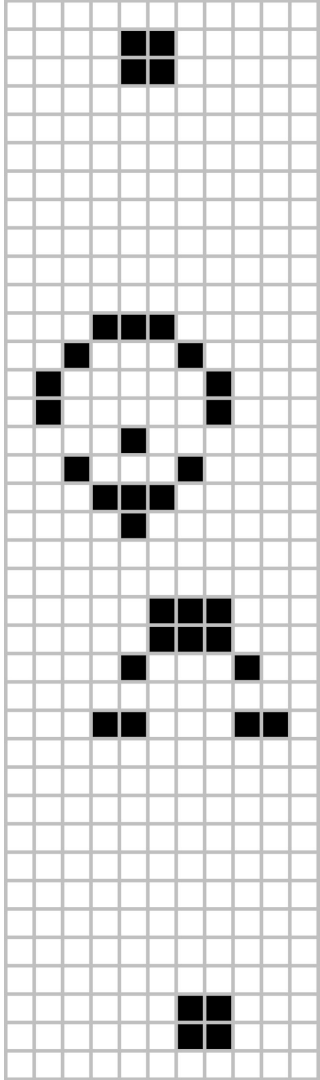
Server-side Kotlin with Coroutines
https://youtu.be/hQrFfwT1IMo

Kotlin Coroutines in Practice
https://youtu.be/a3agLJQ6vt8

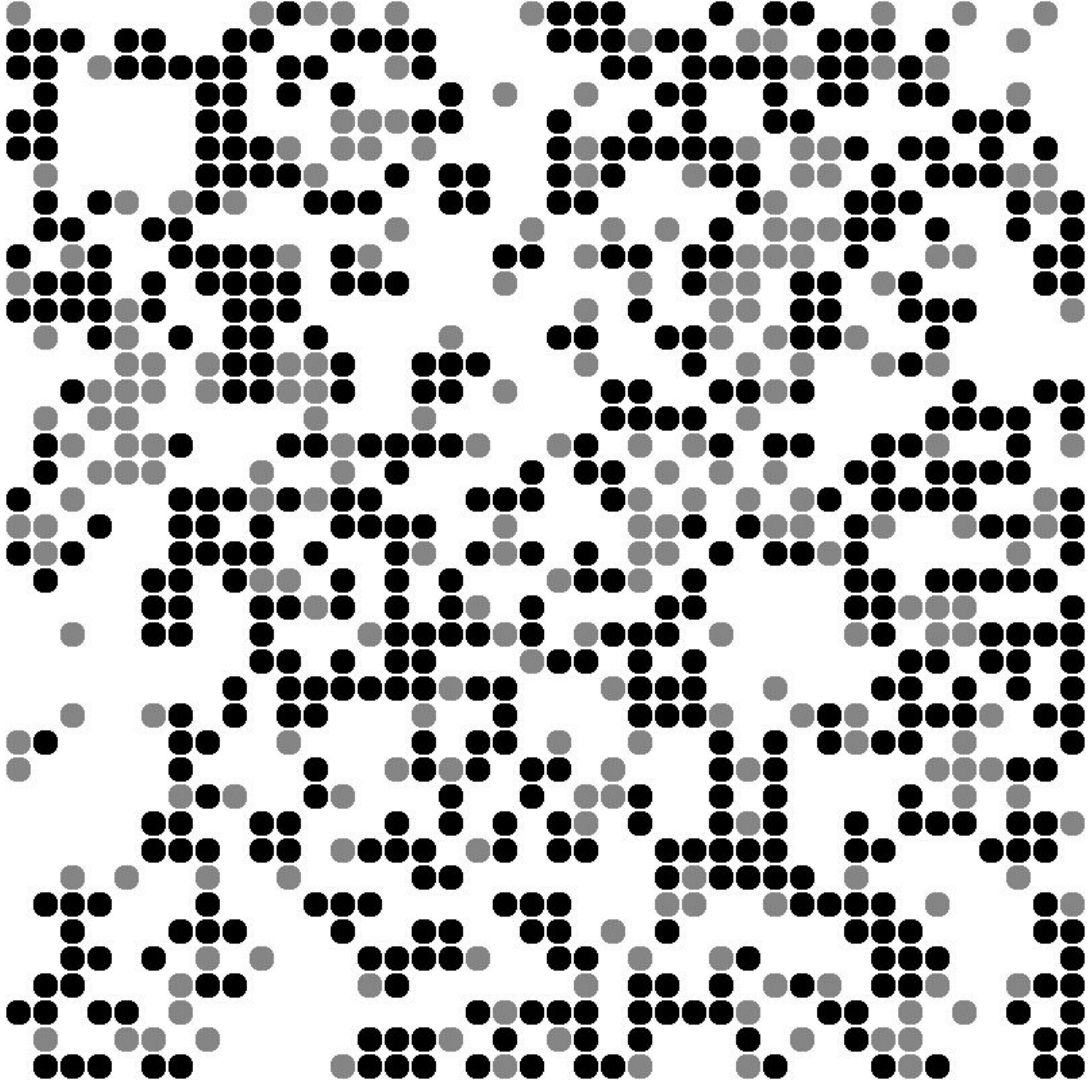
Build with Kotlin

Conway's Game of Life

- Cells in 2D array
- Simple Evolution Rules
- How does it behave?
  - Given starting layout
  - Will someone survive?
  - A periodic evolution?



https://commons.wikimedia.org/wiki/File:Game_of_life_glider_gun.svg



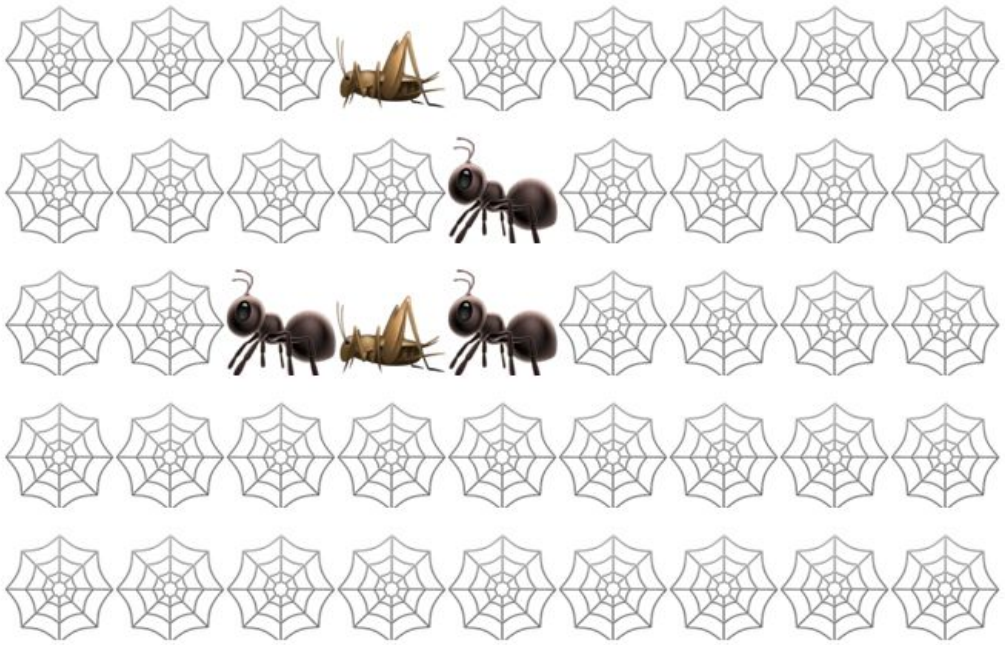
Evolution Rules

```
fun EvolutionCell.conwayLaws() = when (state) {
  ALIVE -> when (neighbours) {
    2, 3 -> ALIVE  // living on
    else -> DEAD   // underpopulation or overpopulation
  }

  DEAD -> when (neighbours) {
    3 -> ALIVE     // reproduction
    else -> DEAD
  }
}
```

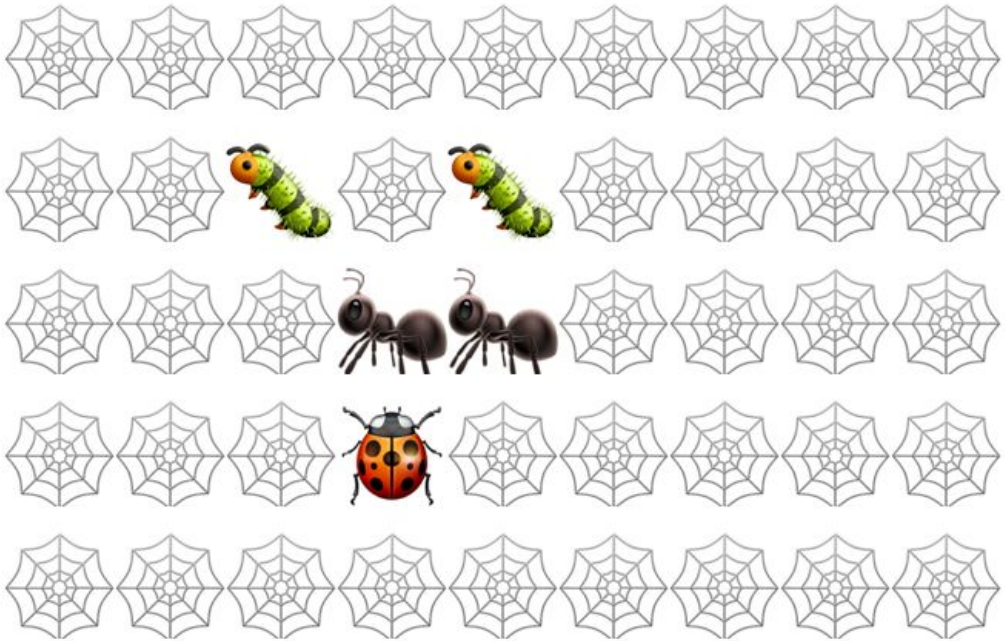
Iteration Code in Console

```
var m = """
    ...x.....
    ....x....
    ..xxx....
    .........
    .........
""".toMaze()

repeat(5) {
  println(m.renderToString())
  m = m.nextGeneration(EvolutionCell::conwayLaws)
}
```

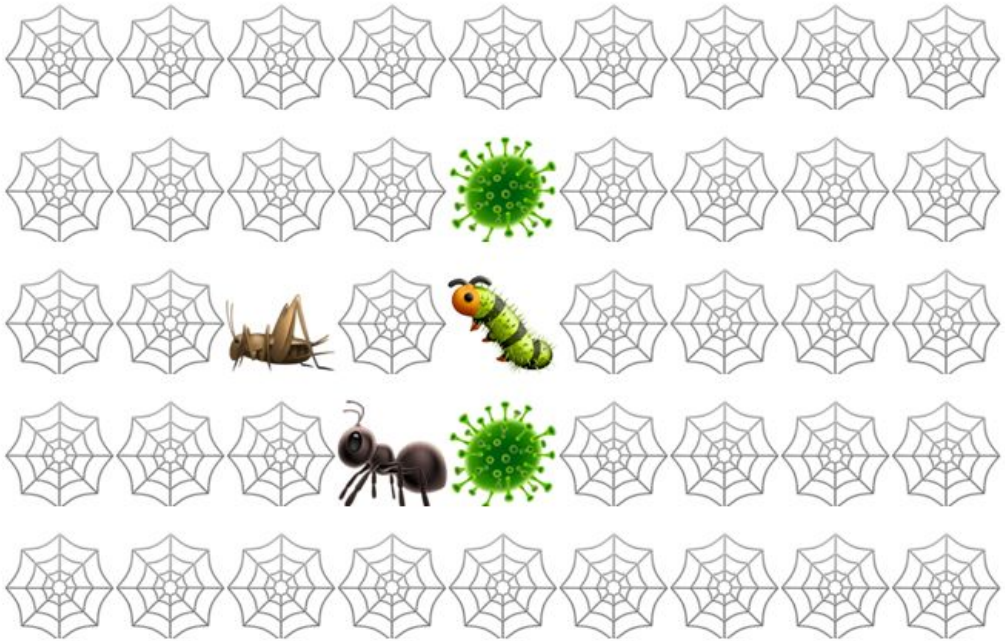
Console Output in Unicode



Console Output in Unicode



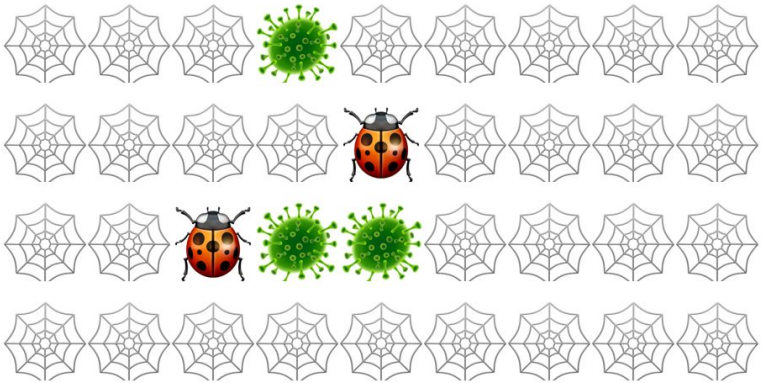
Console Output in Unicode



Kotlin/Native

Console App with Kotlin/Native

```
> Task :runReleaseExecutableConsole
```



```
➜  releaseExecutable git:(master) ✗ l
total 2128
drwxr-xr-x  3 jonnyzzz  staff    96B Jun  3 22:06 .
drwxr-xr-x  4 jonnyzzz  staff   128B Jun  3 22:05 ..
-rwxr-xr-x  1 jonnyzzz  staff   1.0M Jun  3 22:06 kotlin-game-of-life.kexe
```

Kotlin/Native

- Included into
  Kotlin releases as Beta
- LLVM based
- **No Virtual Machine!**

Kotlin/Native Tooling

- Executable
- Static, Dynamic, Shared libraries
- C, Objective-C/Swift Frameworks
- klib

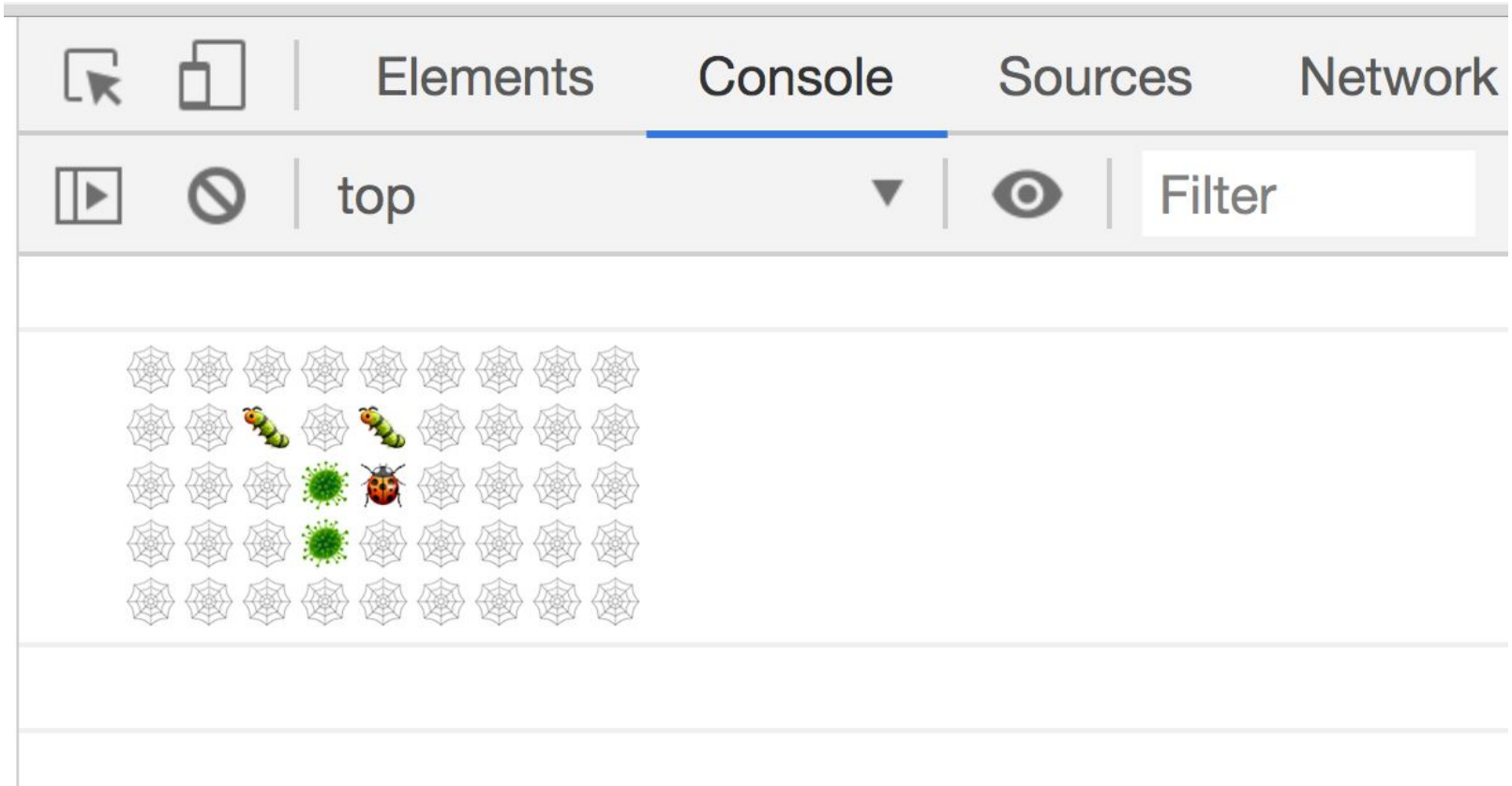
Consoles does not like Unicode

Running in Browser!

- Switch to Kotlin/Common
  - stdlib was enough
- Compile to
  - JS (for browser & console)
  - JVM (for console)

Kotlin/JS

JavaScript with Kotlin/JS

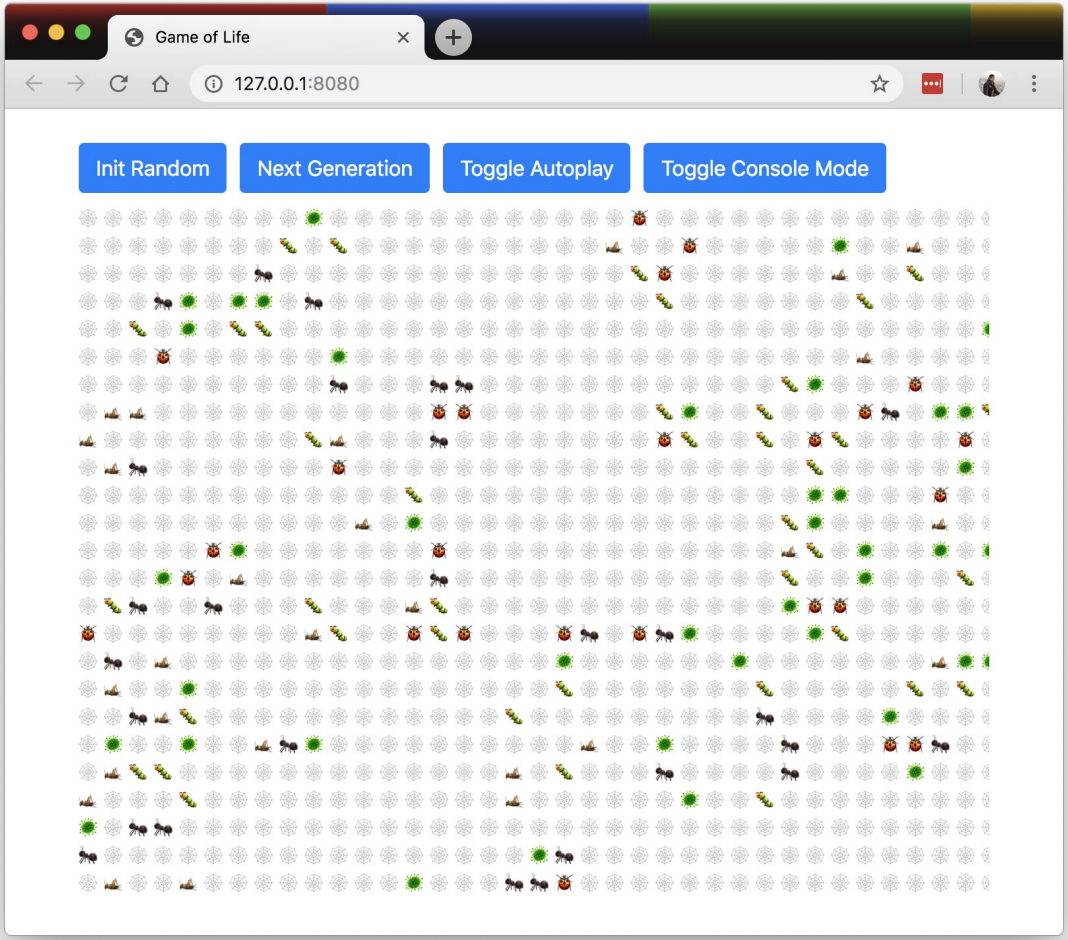


HTML Canvas

- Kotlin/JS code
- Render HTML via DSL
- HTML Canvas

HTML DSL in Kotlin

```
div(classes = "controls btn-block") {
  button {
    +"Init Random"
    classes = setOf("btn", "btn-primary")
    onClickFunction = { world = randomMaze(40, 40) }
  }

  button {
    +"Next Generation"
    classes = setOf("btn", "btn-primary")
    onClickFunction = { nextStep() }
  }

  button {
    +"Toggle Autoplay"
    classes = setOf("btn", "btn-primary")
    onClickFunction = { toggleAutoplay() }
```

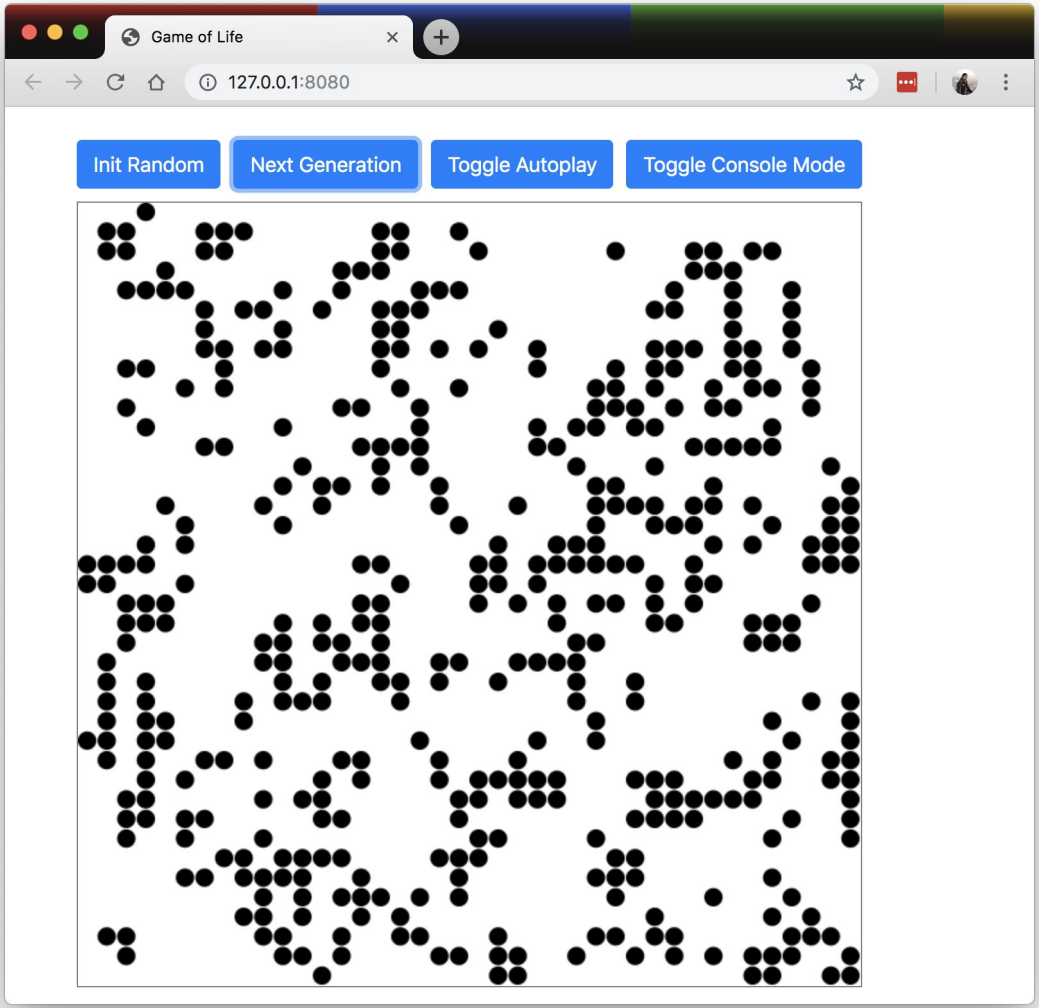

Game of Life
127.0.0.1:8080
Init Random
Next Generation
Toggle Autoplay
Toggle Console Mode

Kotin/JS and Canvas

```
val width = canvas.width.toDouble()
val height = canvas.height.toDouble()

val stepX = width / maze.width
val stepY = height / maze.height

val rX = (stepX - 1) / 2
val rY = (stepY - 1) / 2

with(canvas.getContext("2d") as CanvasRenderingContext2D) {
  clearRect(0.0, 0.0, width, height)
  maze.forEachAlive { x, y ->
    beginPath()
    ellipse(x * stepX + rX, y * stepY + rY, rX, rY, 0.0, 0.0, 2 * PI)
    fill()
  }
}
```

Kotin/JS and Canvas

```
val width = canvas.width.toDouble()
val height = canvas.height.toDouble()

val stepX = width / maze.width
val stepY = height / maze.height

val rX = (stepX - 1) / 2
val rY = (stepY - 1) / 2

with(canvas.getContext("2d") as CanvasRenderingContext2D) {
  clearRect(0.0, 0.0, width, height)
  maze.forEachAlive { x, y ->
    beginPath()
    ellipse(x * stepX + rX, y * stepY + rY, rX, rY, 0.0, 0.0, 2 * PI)
    fill()
  }
}
```


Game of Life
127.0.0.1:8080
Init Random
Next Generation
Toggle Autoplay
Toggle Console Mode

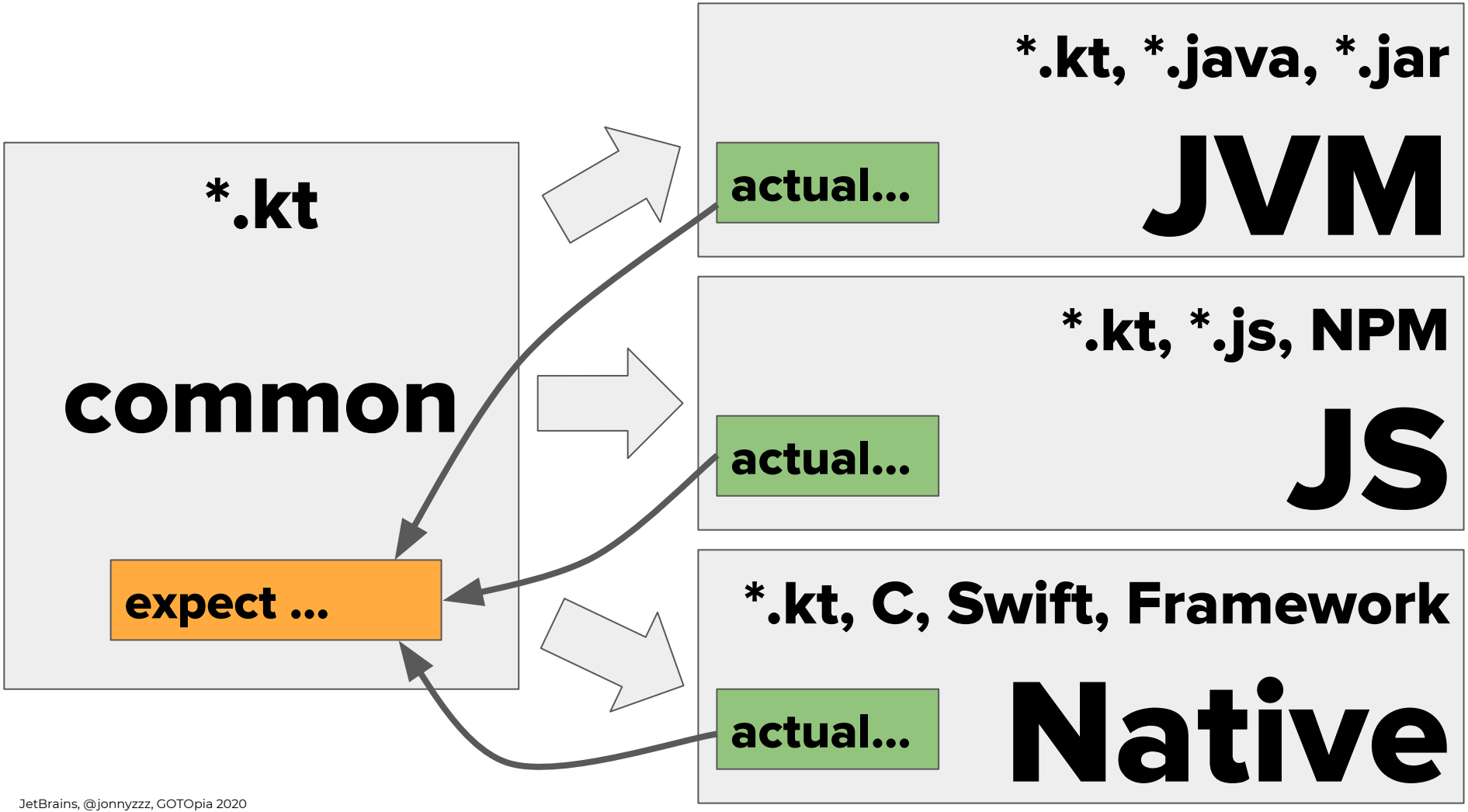
Mobile App

Mobile Support?

- iOS app with Kotlin/Native & Xcode
- Android with Kotlin/JVM

Android Robot - Wikipedia https://en.wikipedia.org/wiki/File:Android_robot.svg
Apple Logo - Wikipedia https://en.wikipedia.org/wiki/File:Apple_logo_black.svg

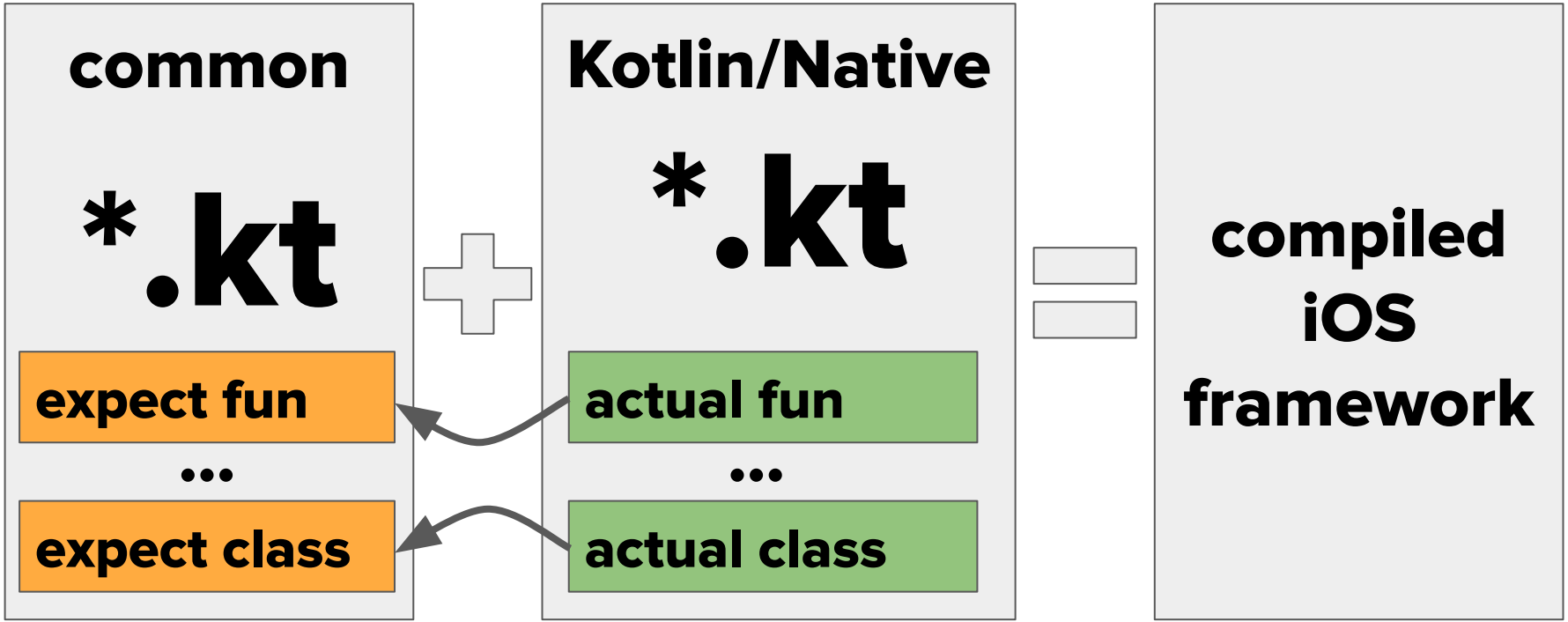
Kotlin/Native

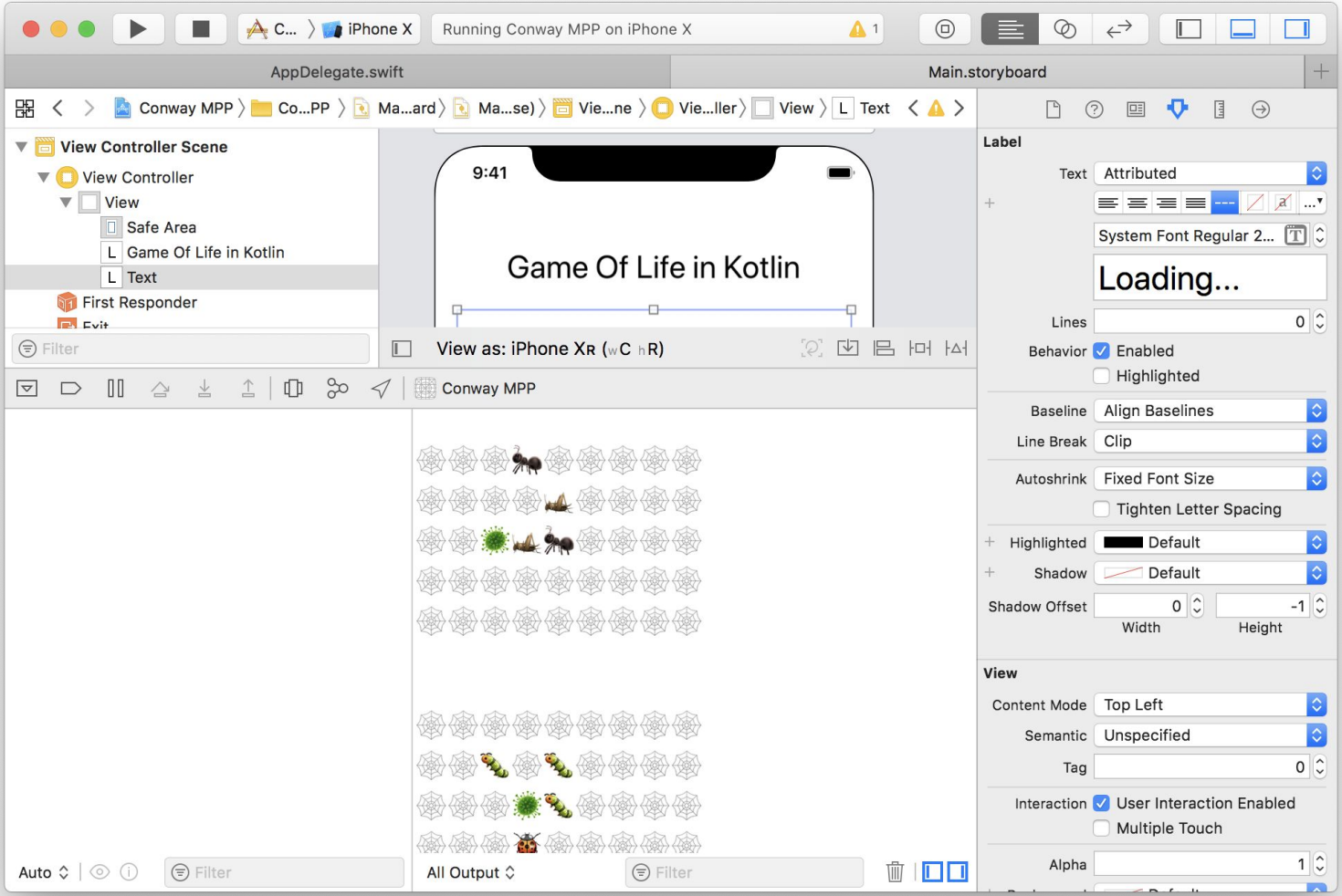

*.kt
common
expect ...
*.kt, *.java, *.jar
actual...
JVM
*.kt, *.js, NPM
actual...
JS
*.kt, C, Swift, Framework
actual...
Native

iOS How To

- Compile Kotlin code as Objective-C/Swift Framework
- Call Kotlin Classes from Swift

iOS with Kotlin




iPhone X
Running Conway MPP on iPhone X
AppDelegate.swift
Main.storyboard
Conway MPP
Co...PP
Ma...ard
Ma...se)
Vie...ne
Vie...ller
View
Text
View Controller Scene
View Controller
View
Safe Area
Game Of Life in Kotlin
Text
First Responder
9:41
Game Of Life in Kotlin
View as: iPhone XR (wC hR)
Label
Text
Attributed
System Font Regular 2...
Loading...
Lines
0
Behavior
Enabled
Highlighted
Baseline
Align Baselines
Line Break
Clip
Autoshrink
Fixed Font Size
Tighten Letter Spacing
Highlighted
Default
Shadow
Default
Shadow Offset
0
-1
Width
Height
View
Content Mode
Top Left
Semantic
Unspecified
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Auto
Filter
All Output

iOS Rendering

- UILabel
- Unicode Emoji
  - 🕸, 🐜, 🐛, 🦗, 🐞, 🪱
- Same Kotlin code

Kotlin + Swift Interop!

```
class WorldWrapper(
  val onWorldChange: (String) -> Unit
) {
```

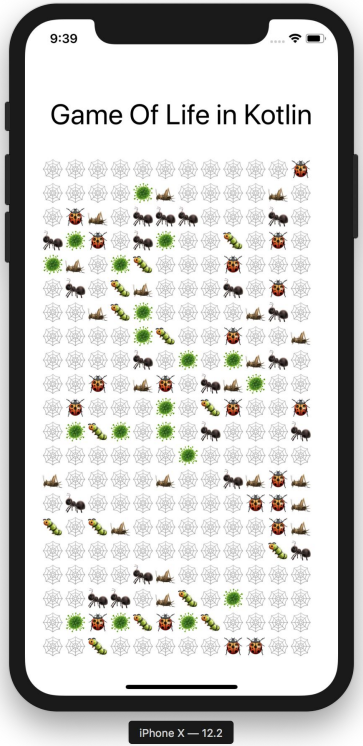
```
world = WorldWrapper { [text] render in
    //TODO: how many chars fits into a line?
    var lineCount = 0
    var newText = ""

    for line in render.split(separator: "\n") {
        newText += line.prefix(12) + " \n"
        lineCount += 1
    }

    text!.numberOfLines = lineCount
    text!.text = newText
}
```

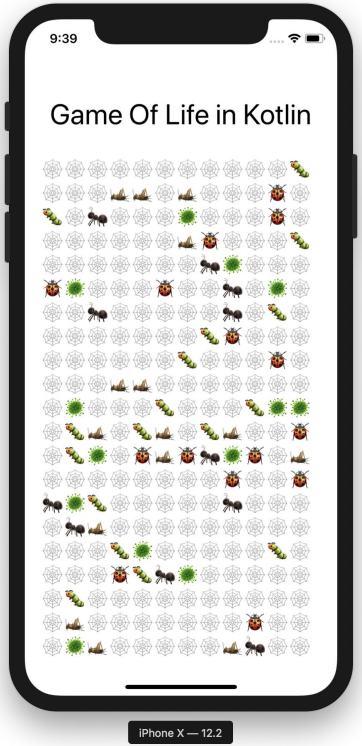
Kotlin + Swift & Xcode

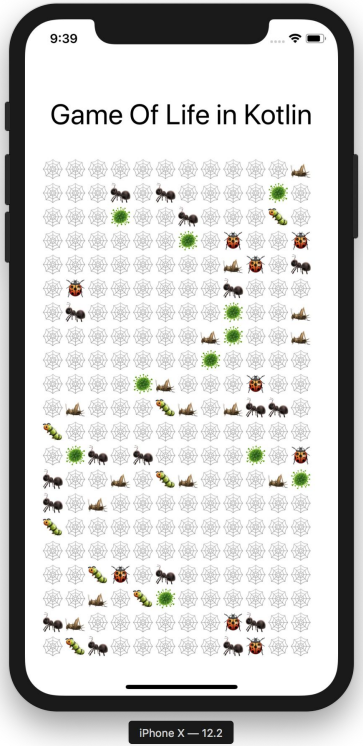
```kotlin
class WorldWrapper(
  val onWorldChange: (String) -> Unit
) {
  private fun initIOS() = randomMaze(40, 40)

  private var world by Delegates.observable(initIOS()) {
    onWorldChange(it.renderToString())
  }

  fun initWorld() {
    world = initIOS()
  }

  fun iterateWorld() {
    world = world.nextGeneration(EvolutionCell::conwayLaws)
  }
}
```

```swift
@IBOutlet weak var text: UILabel!
var world : WorldWrapper!

override func viewDidLoad() {
    super.viewDidLoad()

    world = WorldWrapper { [text] render -> KotlinUnit in
        //TODO: how many chars fits into a line?
        var lineCount = 0
        var newText = ""

        for line in render.split(separator: "\n") {
            newText += line.prefix(12) + "\n "
            lineCount += 1
        }

        text!.numberOfLines = lineCount
        text!.text = newText

        return KotlinUnit()
    }
    world?.doInitWorld()
```

iOS App - same code







An Animated Gif

- Use Kotlin/JVM
- Render via Java API
- Save as Gif
- Serve as HTTP server via Ktor.io

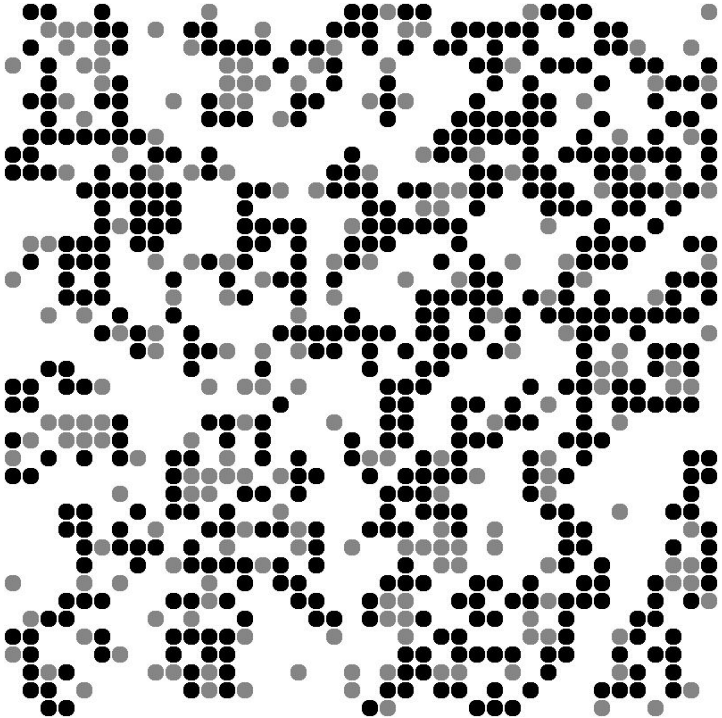
Ktor

Rendering in Java AWT

```
val stepX = image.width / maze.width
val stepY = image.height / maze.height

val ctx = image.graphics
ctx.color = Color.BLACK
maze.forEachAlive { x, y ->
  ctx.fillRoundRect(x * stepX, y * stepY, stepX - 1, stepY - 1, stepX, stepY)
}
```

Gif Generator Tool

```
val images = sequence {
  var world = randomMaze(40, 40)
  var prevImage = world.toImage(800, 800)

  repeat(300) {
    println("Step $it...")
    world = world.nextGeneration(EvolutionCell::conwayLaws)
    prevImage = world.renderToImage(prevImage.addAging())
    yield(prevImage)
  }
}

FileOutputStream(file).use {
  MemoryCacheImageOutputStream(it).use { os ->
    createGIF(os, images)
  }
}
```



Kotlin Multiplatform Mobile

- Known as **KMM**, K/N, KMP, MPP
- JS, JVM, Native, Android, iOS, Wasm
- Example
  - github.com/jonnyzzz/kotlin-game-of-life

https://www.ketchupy.pl/

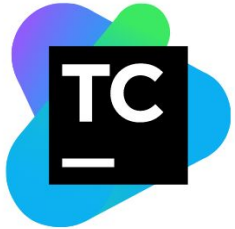
Kotlin for Server-Side & JVM

- Ktor.io - an asynchronous framework for creating microservices, web applications
- Kotless - framework **for serverless** apps
- SpringFramework
- … and the whole JVM ecosystem

Multiplatform Projects

At JetBrains

- We use Kotlin for JVM
  - IntelliJ-based IDEs, tools
  - Servers (TeamCity, YouTrack, ...)
- We use Kotlin Multiplatform
  - Space (Web, Server, Mobile)




DG

PS

RM

TC


PC

CL

IJ

YT

UP

R#

RD

GO

WS

AC


icpc.foundation
icpc
International Collegiate
Programming Contest
JET BRAINS
icpc global
programming
tools sponsor

Kotlin 1.4: One for All

Join the Online Event →
October 12–15

expect fun

actual fun

Thank you!

@jonnyzzz