
LUCAS BORSATTO
Kotlin
Backend

GITHUB.COM/LUCASBSIMAO

SLIDESHARE.NET/LUCASBORSATTO

Conceitos

1

CÓDIGO

Quanto menos melhor:

Enxutamento no número de linhas

Conceitos

2 Extensions

Extension functions:
Helpers, DSLs e lambda receivers

Conceitos

3

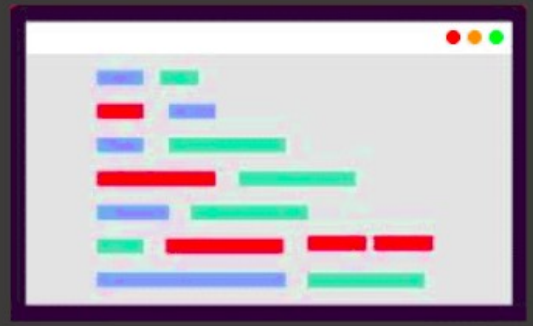
COROUTINES

Concorrência facilitada:
Light-weight threads

Código 1

JAVA

40% OFF



KOTLIN

Código 1

POJO

Código 1

ONE LINE FUNCTIONS

Código 1

CHECKED EXCEPTIONS

Extensions 2

```
fun Date.isATuesday(): Boolean{
  return day == 2
}
```

```
fun Date.isATuesday: Boolean
  get() = day == 2
```

Extensions 2

```
fun execute(body: (EditText) -> {
   body(EditText()) //passando como parâmetro
}

execute {
   text = "" // não funciona
   it.text = "teste" // atribuição normal
}
```

Extensions 2

```
fun execute(body: EditText.() -> {
   EditText().body() //executando como uma extension func
}

execute {
   this.text = "teste"
   text = "teste" // mesmo efeito que acima
   it.text = "" // não funciona
}
```

Extensions 2

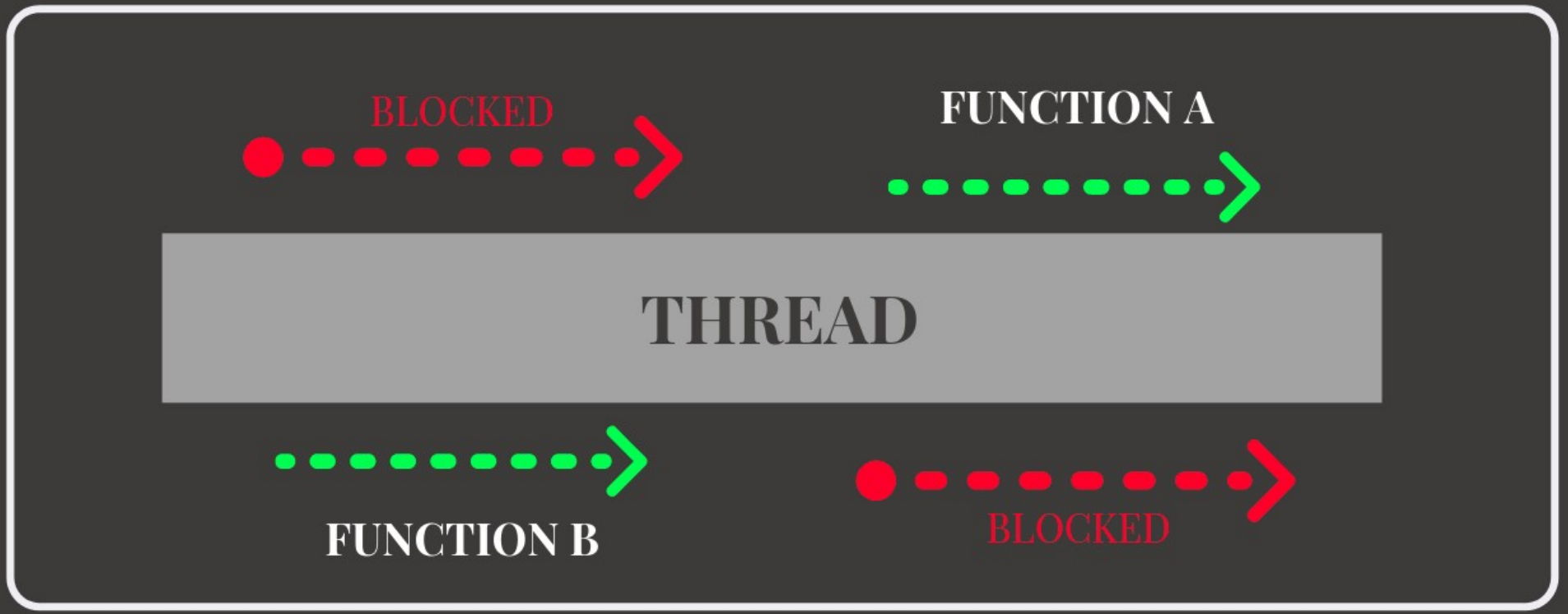
exemplos

Anko

Ktor

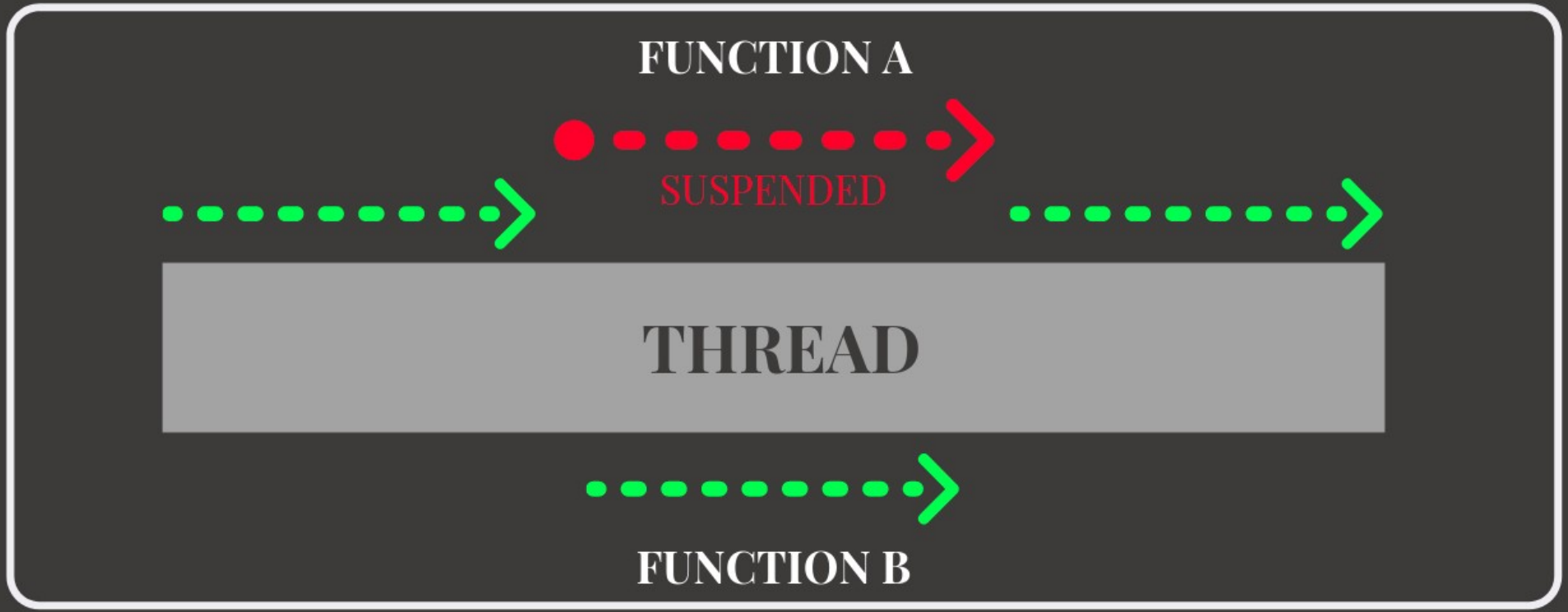
Coroutines 3

execução de threads



Coroutines 3

execução de coroutines



Coroutines 3

operações bloqueantes

1. CPU {

```
fun findBigPrime(): BigInteger =
    BigInteger.probablePrime(4096, Random())
```

2. IO {

```
fun BufferedReader.readMessage(): Message? =
    readLine()?.parseMessage()
```

Coroutines 3

```
import kotlinx.coroutines.*

fun main(): runBlocking<Unit> {
    launch {
        delay(1000L)
        println("World")
    }
    println("Hello")
    delay(2000L)
}
```

Coroutines 3

```
import kotlinx.coroutines.*

0. fun main(): runBlocking<Unit> {
    launch {
        delay(1000L)
        println("World")
    }
    println("Hello")
    delay(2000L)
}
```

Coroutines 3

```
import kotlinx.coroutines.*

0. fun main(): runBlocking<Unit> {
   1. launch {
        delay(1000L)
        println("World")
      }
      println("Hello")
      delay(2000L)
}
```

Coroutines 3

```
import kotlinx.coroutines.*

0. fun main(): runBlocking<Unit> {
   1. launch {
      2. delay(1000L)
      println("World")
   }
   println("Hello")
   delay(2000L)
}
```

Coroutines 3

```
import kotlinx.coroutines.*

0. fun main(): runBlocking<Unit> {
   1. launch {
      2. delay(1000L)
      println("World")
   }
   3. println("Hello")
   delay(2000L)
}
```

Coroutines 3

```
import kotlinx.coroutines.*

0. fun main(): runBlocking<Unit> {
   1. launch {
      2. delay(1000L)
      println("World")
   }
   3. println("Hello")
   4. delay(2000L)
}
```

Coroutines 3

```
import kotlinx.coroutines.*

0. fun main(): runBlocking<Unit> {
   1. launch {
      2. delay(1000L)
      5. println("World")
   }
   3. println("Hello")
   4. delay(2000L)
}
```

Créditos

- Kotlin Talk - Roman Elizarov:
https://www.youtube.com/watch?v=_hfBv0a09Jc
- Podcast Hipsters :
https://hipsters.tech/kotlin-android-e-alem-hipsters-110/
- Docs:
https://kotlinlang.org/docs/reference/

Obrigado!