Creating Coroutines

Douglas Starnes
Author / Speaker

@poweredbyalnet douglasstarnes.com

Kotlin Coroutines

Coroutines process background tasks without the overhead of multithreading

Coroutine builders

Suspend functions

Coroutine scopes

Coroutine contexts

Structured concurrency

Cooperative code

Kotlin coroutines are the future!

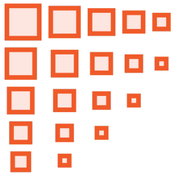
AsyncTask

Memory leaks

Multithreading complexity

Steep learning curve

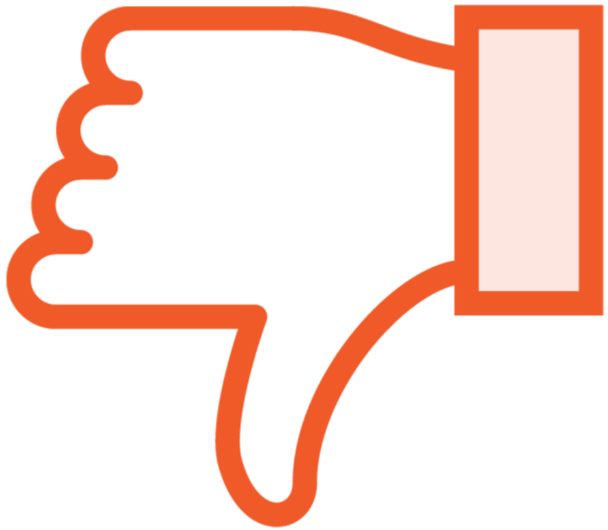
Not good for longer tasks



Deprecated in Android 11 (API Level 30)

Executor

Offers some improvements



Doesn’t avoid multithreading and the associated complexity

Meet Kotlin Coroutines

Easier to avoid problems, such as memory leaks

Simpler API

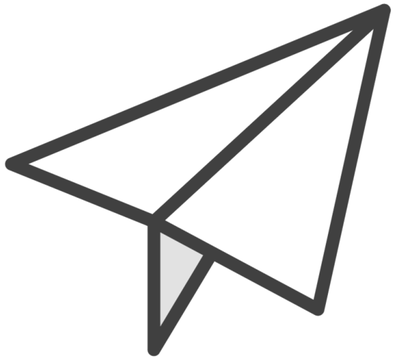
Simpler error handling

Coroutines are not threads

Coroutines can run on the main thread, without blocking!

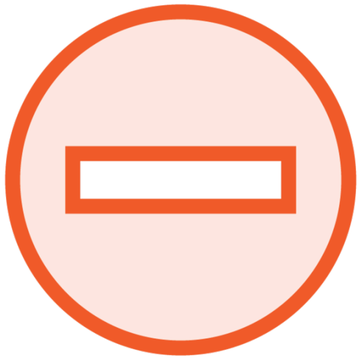
Coroutine Builders

As the name suggests, coroutine builders create a new coroutine.



`async`

Creates a coroutine that returns a value



`launch`

Creates a coroutine that does not return a value

`runBlocking`

For now, think of it as a sandbox for coroutines

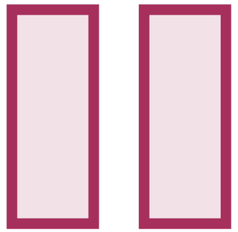
Coroutine Builders

```
fun main() {
  runBlocking {
    launch {
      println("I'm running inside a coroutine!")
    }
  }
}
```

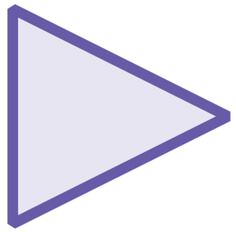
Kotlin Coroutines

Kotlin coroutines are not threads

Kotlin coroutines are an ‘instance of suspendable computation’



Suspending a coroutines, pauses it, without blocking the thread



Coroutines are later resumed and continue to run

Suspend Functions

```
fun main() {
  runBlocking {
    launch {
      delay(500)
      println(“I’m running inside a coroutine”)
    }
  }
}
```

Suspend Functions

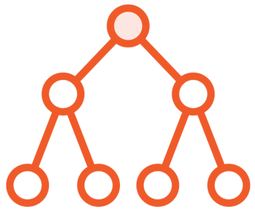
```
fun main() {
  runBlocking {
    launch {
      mySuspendableFunction()
    }
  }
}

suspend fun mySuspendableFunction() {
  delay(500)
  println("I'm running inside a coroutine")
}
```

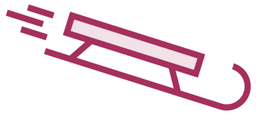
Cooperative code

Kotlin coroutines are not a feature of Android

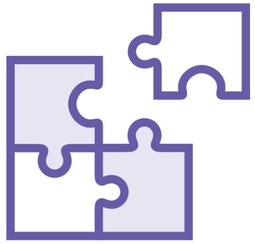
Structured Concurrency



Coroutines arranged in parent/child relationships



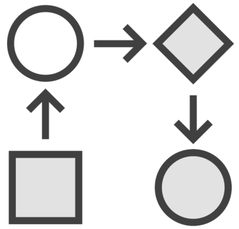
Child coroutines should finish before parents



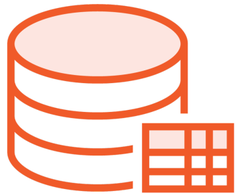
Prevents orphaned coroutines

CoroutineScope

GlobalScope



LifecycleScope



viewModelScope

Coroutine scopes manage the lifecycle of coroutines

CoroutineContext

Define the environment for a `CoroutineScope`

A `Job` **is a reference to a** `CoroutineScope` **(or coroutine)**

A `Dispatcher` **is the type of thread the** `CoroutineScope` **will run in**

Dispatchers

Default

IO

Main

Unconfined

CoroutineScope

```
val scope = CoroutineScope(Dispatchers.Default)
```

CoroutineScope

```
val scopeJob = Job()
val scope = CoroutineScope(Dispatchers.Default)
```

CoroutineScope

```
val scopeJob = Job()
val scope = CoroutineScope(Dispatchers.Default + scopeJob)
```

CoroutineScope

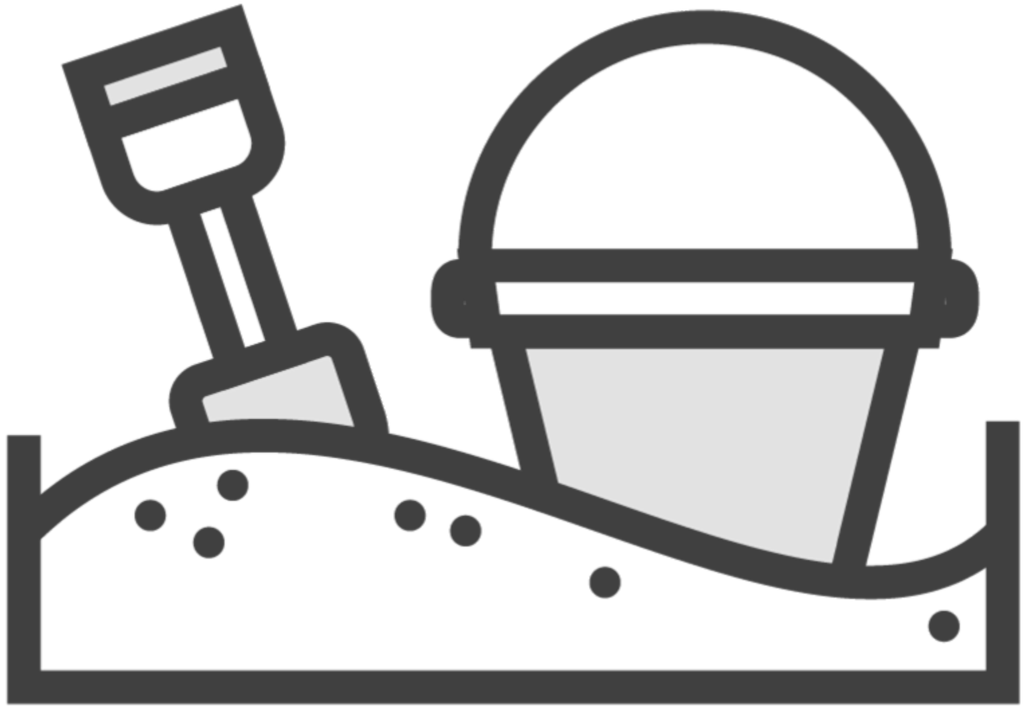
```
val scopeJob = Job()
val scope = CoroutineScope(Dispatchers.Default + scopeJob)
val coroutineJob = scope.launch {
  delay(500)
  println("I'm running inside of a coroutine with a custom CoroutineContext")
}
```

`CoroutineScope` Builders

`coroutineScope`

`withContext`

`runBlocking` as a Coroutine Container



Use Cases for `runBlocking`

A bridge between blocking code and coroutines

Unit testing

`runBlocking` in Console Apps

What in the world are coroutines?

```
fun main() {

}
```

`runBlocking` in Console Apps

```
fun main() {
  runBlocking {



  }
}
```

Don't worry about it, I can handle them.

`runBlocking` in Console Apps

Go for it!

```
fun main() {
  runBlocking {
    launch {
      delay(...)
    }
  }
}
```

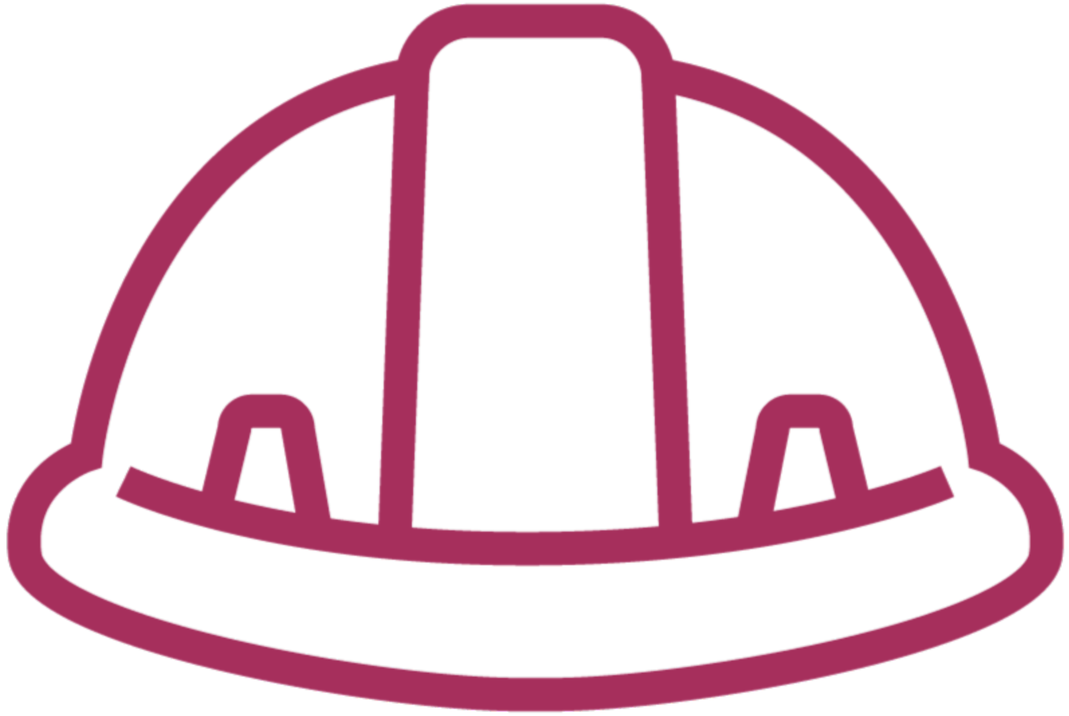
GlobalScope

```
fun main() {
  GlobalScope.launch {
    delay(...)
    println(...)
  }
}
```

GlobalScope

```
fun main() {
  GlobalScope.launch {
    delay(...)
    println(...)
  }
  Thread.sleep(...)
}
```

`GlobalScope` Is Dangerous



Moral of the Story

Don't use `GlobalScope`

Use `runBlocking` **as a bridge between blocking code and coroutines, or in unit tests**

Summary

Kotlin coroutines improve upon the past attempts at Android background processing

Coroutine builders

- `launch` returns no value
- `async` returns a value
- `runBlocking` is for special cases

Cooperative code

- Suspend functions

Summary

Coroutine scopes

- Parent/child relationships
- Manage coroutine lifecycles
- Coroutine contexts, dispatchers, jobs
- Avoid using the `GlobalScope`
- `coroutineScope` builder

Reference coroutines and coroutine scopes with jobs

Kotlin coroutines are not specific to Android