

**Kotlin Coroutines**

**Course Number:** MBL-232
**Duration:** 2 days

**Overview**

[Coroutines](https://developer.android.com/kotlin/coroutines), as implemented in Kotlin, are an elegant way of solving many complex programming problems including asynchronous programming, actors models, lazy evaluated iteration, and more.

In this Kotlin Coroutines training course attendees learn Coroutines fundamentals as well as many of the advanced features, including structured concurrency, cancellation, and more.

**Prerequisites**

All attendees must be familiar with the fundamentals of programming in Kotlin.

**Materials**

All attendees receive comprehensive courseware.

Course outline and materials are copyrighted and owned by [Instil Software](https://instil.co/).

**Software Needed on Each Student PC**

* Windows or Mac minimum 8 GB RAM
* Android Studio installed
* Provided lab files from Accelebrate

**Objectives**

* Understand the purpose and advantages of Coroutines
* Apply coroutines for asynchronous programming
* Make use of all aspects of the Coroutines library
* Explore practical applications of coroutines

**Outline**

* Coroutine Fundamentals
	+ What precisely is a Kotlin Coroutine?
	+ Comparing Coroutines to Java Threads
	+ Comparing Coroutines to the Stream API
	+ How does a suspending function work?
	+ The responsibilities of the compiler
	+ The responsibilities of the library
	+ Potential for multiple implementations
	+ Choosing and using bridge functions
	+ Sequences, channels, and other abstractions
	+ Behavior when limits are met
	+ Commonly applied asynchronous patterns
* Structured Concurrency with Coroutines
	+ A tour of the types within the Coroutines API
	+ The importance of the CoroutineScope and Context
	+ Defining Supervisor Scopes
	+ Using the standard dispatchers and creating your own
	+ Options (including actors) for sharing mutable state
	+ Modeling asynchronous sequences of values as Flows
	+ How different operators are implemented within Flows
	+ Understanding context and termination within Flows
* Coroutine Best Practices
	+ Ensuring your Coroutine based design is fault-tolerant
	+ Managing exceptions in Coroutine code
	+ Avoiding Deadlock with coroutines
	+ Testing and validating coroutines
* Conclusion