

**Introduction to RxJava**

**Course Number:** JAV-420  
**Duration:** 2 days

**Overview**

RxJava is a library for composing asynchronous and event-based programs using observable sequences. This Introduction to RxJava training teaches attendees how to implement RxJava in their applications.

**Prerequisites**

All attendees must have basic knowledge of Java and functional interfaces.

**Materials**

All RxJava training students receive comprehensive courseware.

**Software Needed on Each Student PC**

* An installation of JDK 11+ and your favorite IDE (IntelliJ preferred but not required)
* Maven 3.8.x

**Objectives**

* Develop pipelines from Source to Sink
* Tie in functional operators to perform duties while avoiding callback loops
* Use operators to leverage parallelism and concurrency
* Use hot and cold observables, backpressure, basic functional operators, and forking
* Use schedulers to process information asynchronously
* Incorporate unit testing

**Outline**

* Introduction
  + What is it?
  + Reactive Streams
  + Flow API
  + Differences between 1.x and 2.x
* Basic Components
  + Observable
  + Subscriber
  + Subscription
  + Subject
  + Multiple Subscribers
  + Infinite Streaming
* Marble Diagrams
  + Pipeline Creation
  + just
  + interval
  + fromFuture
  + fromIterable
  + fromCallable
  + defer
* Intermediate Operators
  + filter
  + map
  + flatMap
  + flatMapIterable
  + compose and Tranformer
  + takeWhile
  + concatMap
  + zip and zipWith
* Debugging Operators
  + doOnNext
  + doOnError
  + onErrorReturn
  + onAfterTerminate
* Combination Operators
  + startWith
  + concat
  + amb
  + merge
  + switchOnNext
  + combine
  + combineLatest
  + withLatestFrom
  + amb
* Reduction Operators
  + collect
  + reduce
  + scan
  + distinct
  + groupBy
* Error Handling
  + onErrorResumeNext
  + onErrorReturn
  + onErrorReturnItem
  + onExceptionResumeNext
* Multithreading and Schedulers
  + subscribeOn
  + observeOn
* Hot vs. Cold Observable
  + publish
  + share
* Backpressure
  + Flowable
  + throttle
  + throttleWithTimeout
  + sample
  + debounce
* Testing
* Performance
* Conclusion