

This page lists various utility operators for working with Observables.

- [`materialize\(_\)`](#) — convert an Observable into a list of Notifications
- [`dematerialize\(_\)`](#) — convert a materialized Observable back into its non-materialized form
- [`timestamp\(_\)`](#) — attach a timestamp to every item emitted by an Observable
- [`serialize\(_\)`](#) — force an Observable to make serialized calls and to be well-behaved
- [`cache\(_\)`](#) — remember the sequence of items emitted by the Observable and emit the same sequence to future Subscribers
- [`observeOn\(_\)`](#) — specify on which Scheduler a Subscriber should observe the Observable
- [`subscribeOn\(_\)`](#) — specify which Scheduler an Observable should use when its subscription is invoked
- [`doOnEach\(_\)`](#) — register an action to take whenever an Observable emits an item
- [`doOnNext\(_\)`](#) — register an action to call just before the Observable passes an `onNext` event along to its downstream
- [`doAfterNext\(_\)`](#) — register an action to call after the Observable has passed an `onNext` event along to its downstream
- [`doOnCompleted\(_\)`](#) — register an action to take when an Observable completes successfully
- [`doOnError\(_\)`](#) — register an action to take when an Observable completes with an error
- [`doOnTerminate\(_\)`](#) — register an action to call just before an Observable terminates, either successfully or with an error
- [`doAfterTerminate\(_\)`](#) — register an action to call just after an Observable terminated, either successfully or with an error
- [`doOnSubscribe\(_\)`](#) — register an action to take when an observer subscribes to an Observable
- 1.x [`doOnUnsubscribe\(_\)`](#) — register an action to take when an observer unsubscribes from an Observable
- [`finallyDo\(_\)`](#) — register an action to take when an Observable completes
- [`doFinally\(_\)`](#) — register an action to call when an Observable terminates or it gets disposed
- [`delay\(_\)`](#) — shift the emissions from an Observable forward in time by a specified amount
- [`delaySubscription\(_\)`](#) — hold an Subscriber's subscription request for a specified amount of time before passing it on to the source Observable
- [`timeInterval\(_\)`](#) — emit the time lapsed between consecutive emissions of a source Observable
- [`using\(_\)`](#) — create a disposable resource that has the same lifespan as an Observable
- [`single\(_\)`](#) — if the Observable completes after emitting a single item, return that item, otherwise throw an exception
- [`singleOrDefault\(_\)`](#) — if the Observable completes after emitting a single item, return that item, otherwise return a default item
- [`repeat\(_\)`](#) — create an Observable that emits a particular item or sequence of items repeatedly
- [`repeatWhen\(_\)`](#) — create an Observable that emits a particular item or sequence of items repeatedly, depending on the emissions of a second Observable