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