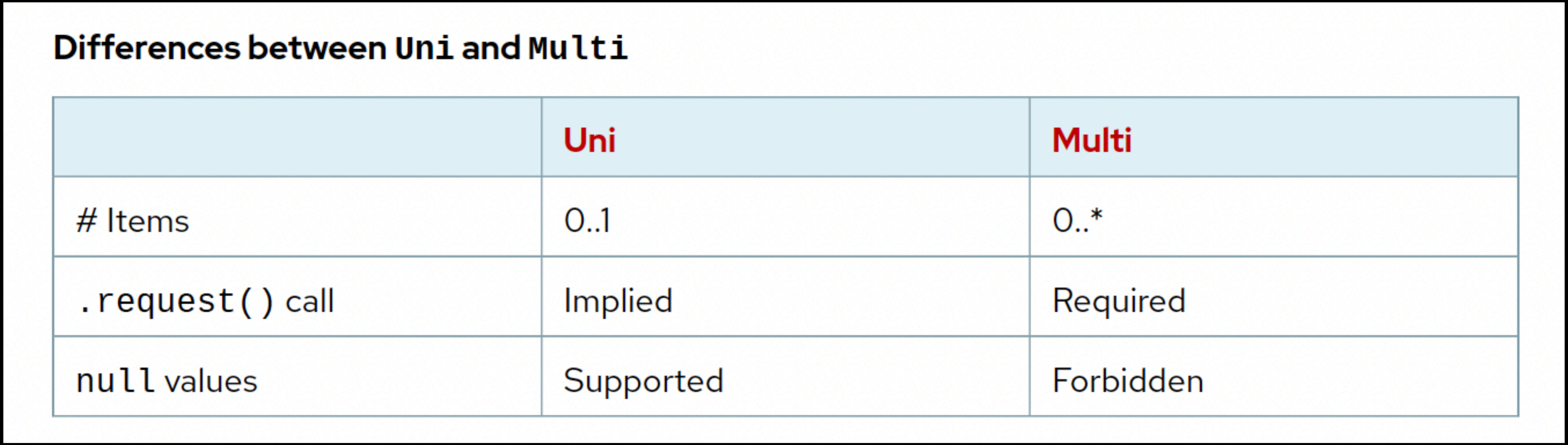
Mutiny

Reactive Library

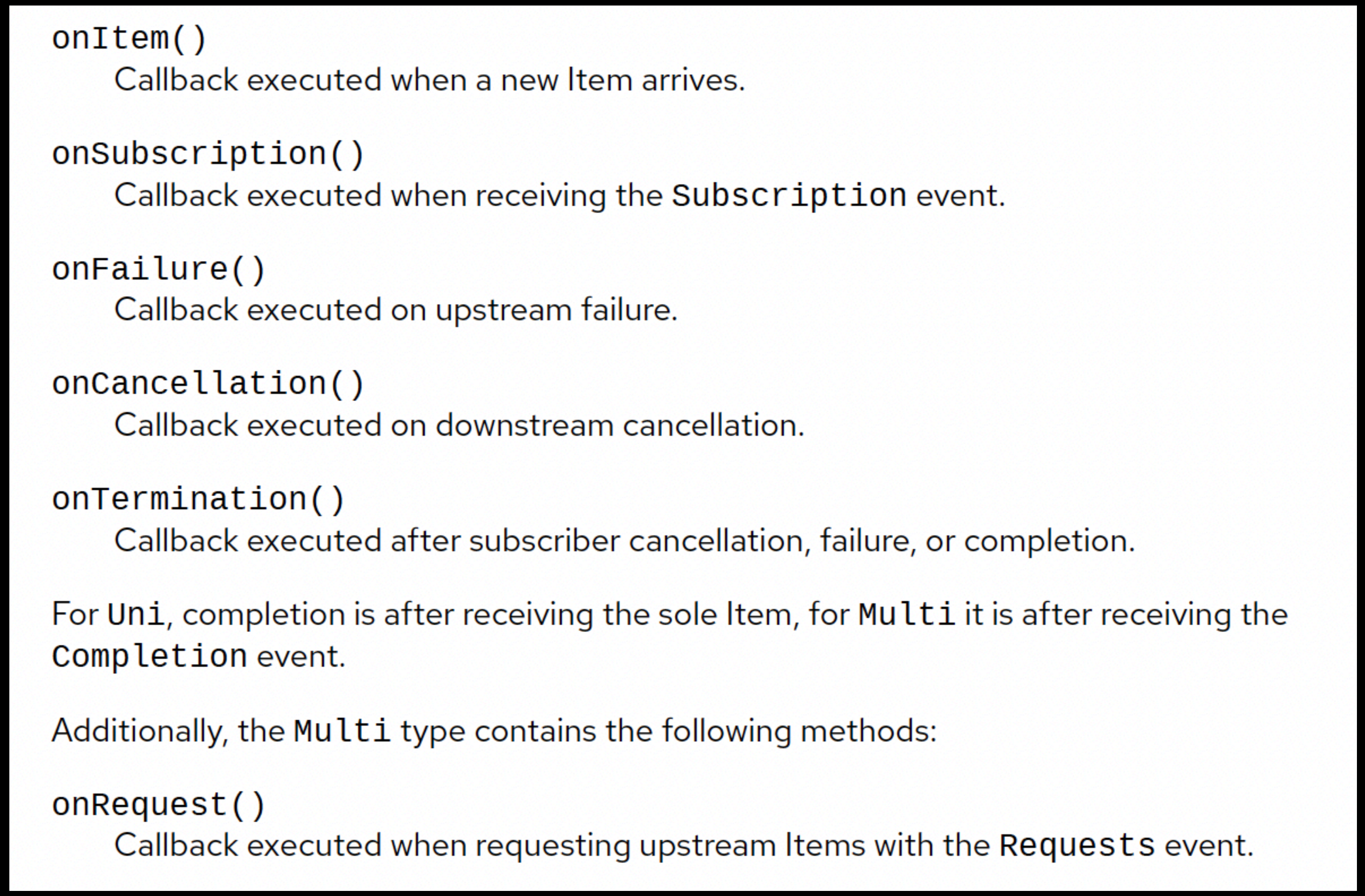
JoeDayz | 2024

io.smallrye.mutiny



Differences between `Uni` and `Multi`

	Uni	Multi
# Items	0..1	0..*
`.request()` call	Implied	Required
`null` values	Supported	Forbidden



`onItem()`
: Callback executed when a new Item arrives.

`onSubscription()`
: Callback executed when receiving the `Subscription` event.

`onFailure()`
: Callback executed on upstream failure.

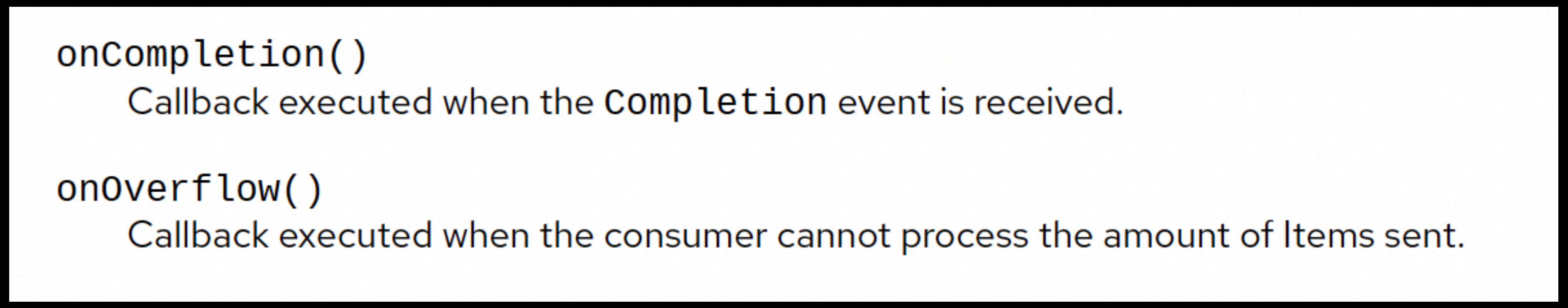
`onCancellation()`
: Callback executed on downstream cancellation.

`onTermination()`
: Callback executed after subscriber cancellation, failure, or completion.

For `Uni`, completion is after receiving the sole Item, for `Multi` it is after receiving the `Completion` event.

Additionally, the `Multi` type contains the following methods:

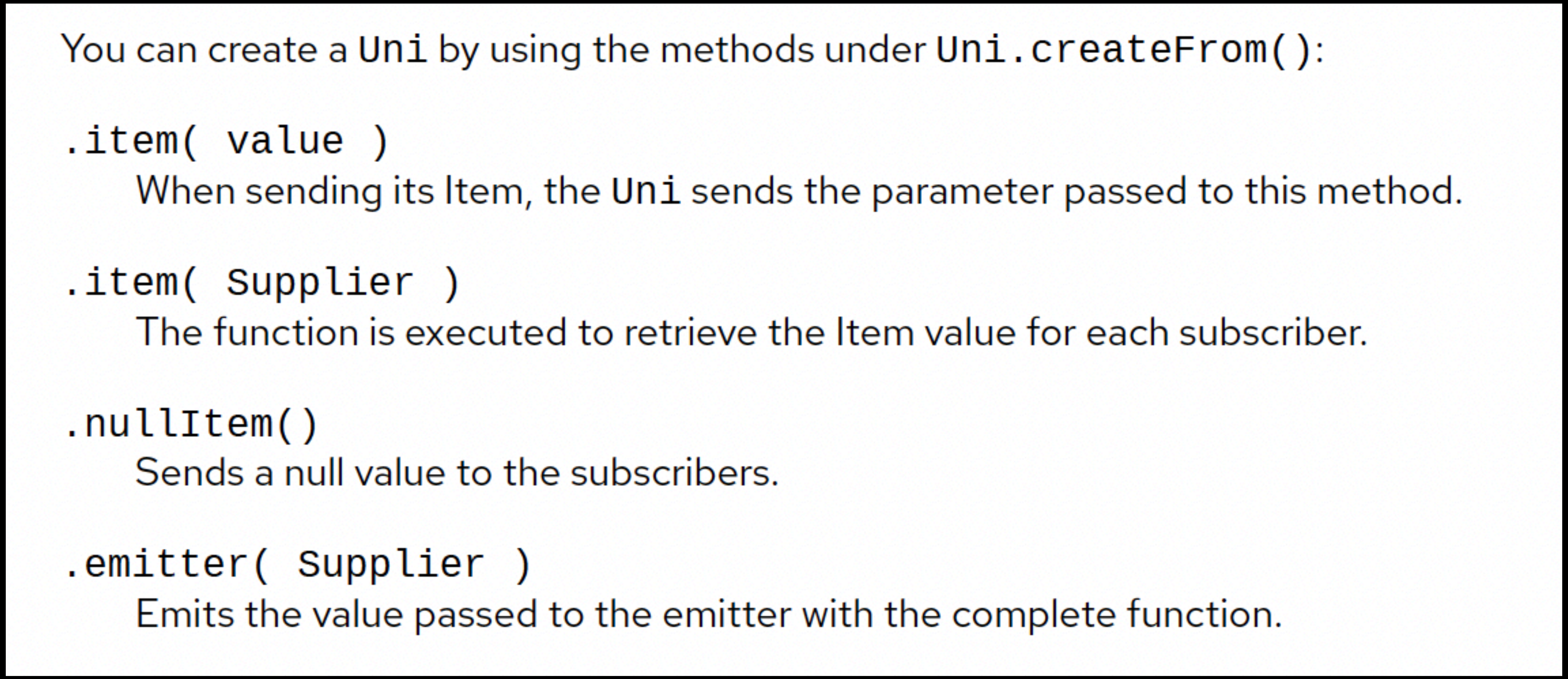
`onRequest()`
: Callback executed when requesting upstream Items with the `Requests` event.



`onCompletion()`
: Callback executed when the `Completion` event is received.

`onOverflow()`
: Callback executed when the consumer cannot process the amount of Items sent.

Crear Uni y Multi



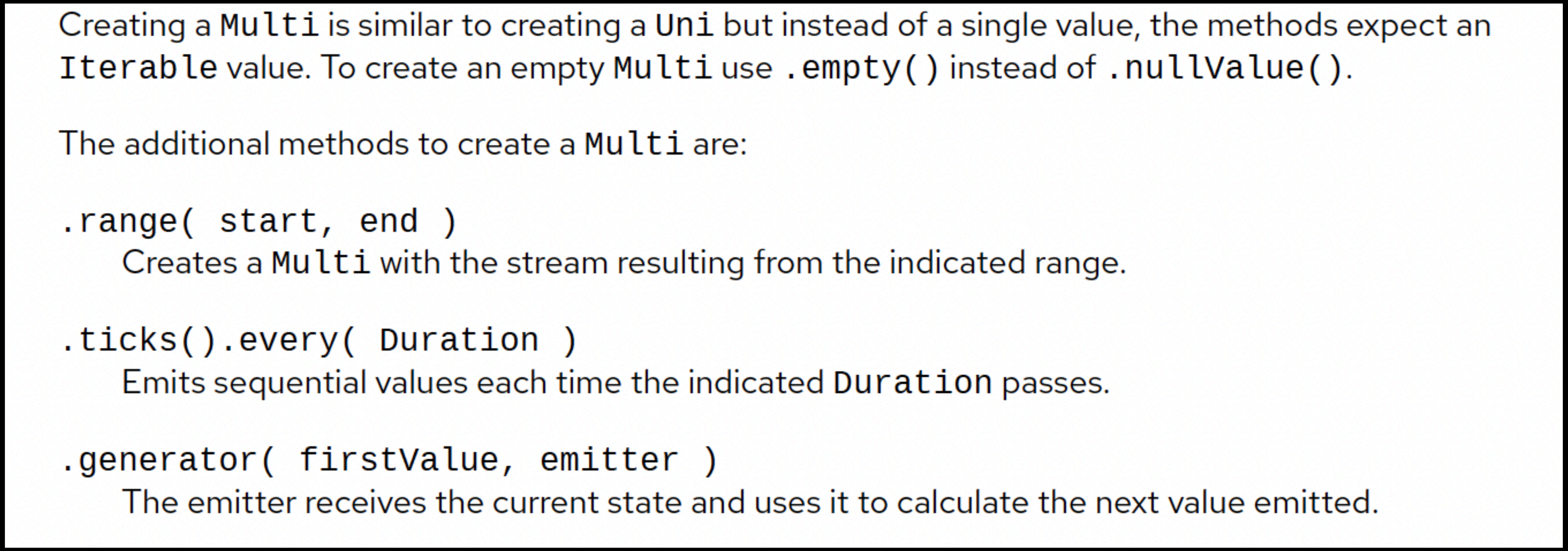
You can create a `Uni` by using the methods under `Uni.createFrom()`:

`.item( value )`
When sending its Item, the `Uni` sends the parameter passed to this method.

`.item( Supplier )`
The function is executed to retrieve the Item value for each subscriber.

`.nullItem()`
Sends a null value to the subscribers.

`.emitter( Supplier )`
Emits the value passed to the emitter with the complete function.



Creating a `Multi` is similar to creating a `Uni` but instead of a single value, the methods expect an `Iterable` value. To create an empty `Multi` use `.empty()` instead of `.nullValue()`.

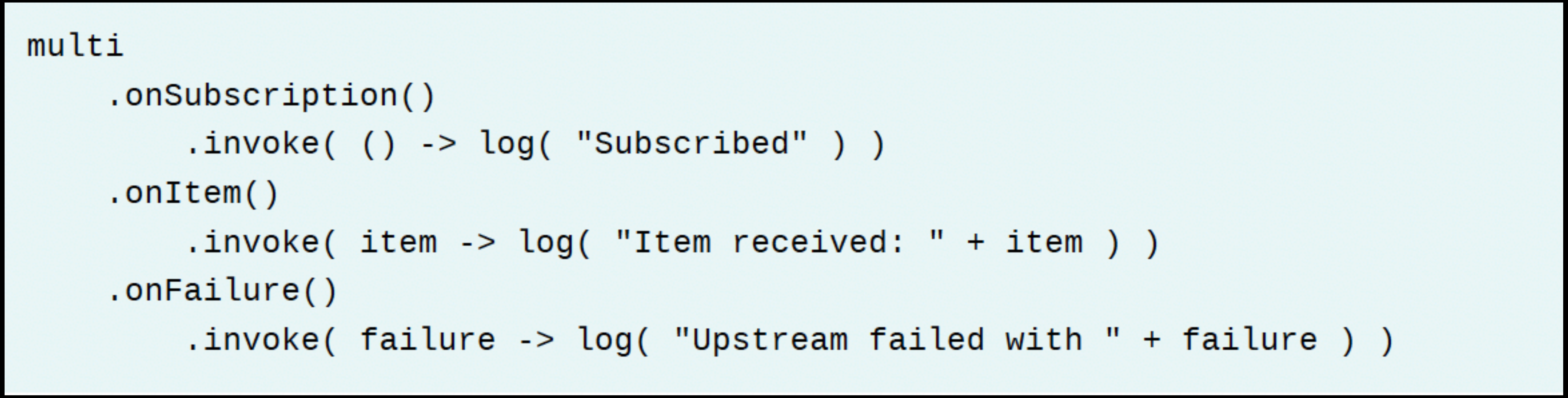
The additional methods to create a `Multi` are:

`.range( start, end )`
Creates a `Multi` with the stream resulting from the indicated range.

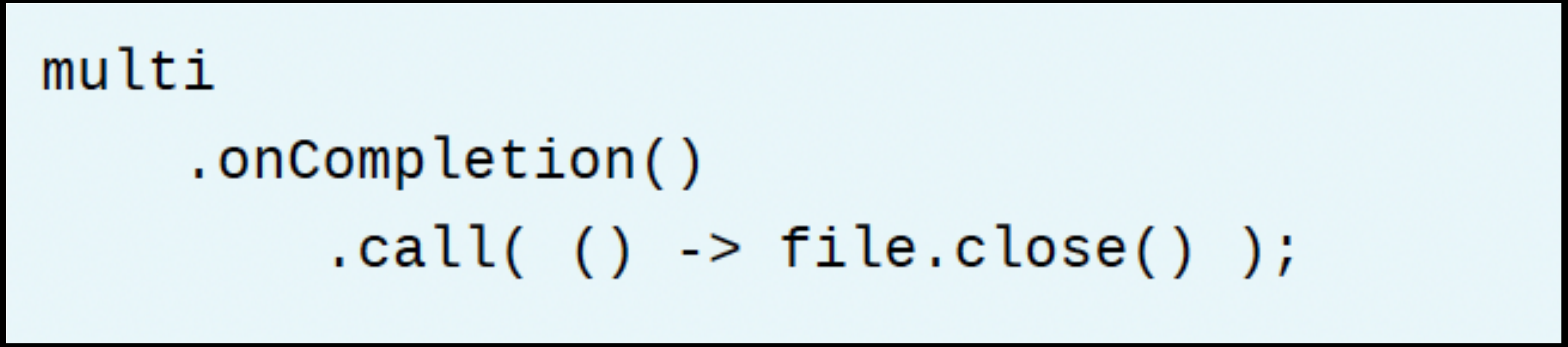
`.ticks().every( Duration )`
Emits sequential values each time the indicated `Duration` passes.

`.generator( firstValue, emitter )`
The emitter receives the current state and uses it to calculate the next value emitted.

Observación de Eventos



```
multi
    .onSubscription()
        .invoke( () -> log( "Subscribed" ) )
    .onItem()
        .invoke( item -> log( "Item received: " + item ) )
    .onFailure()
        .invoke( failure -> log( "Upstream failed with " + failure ) )
```

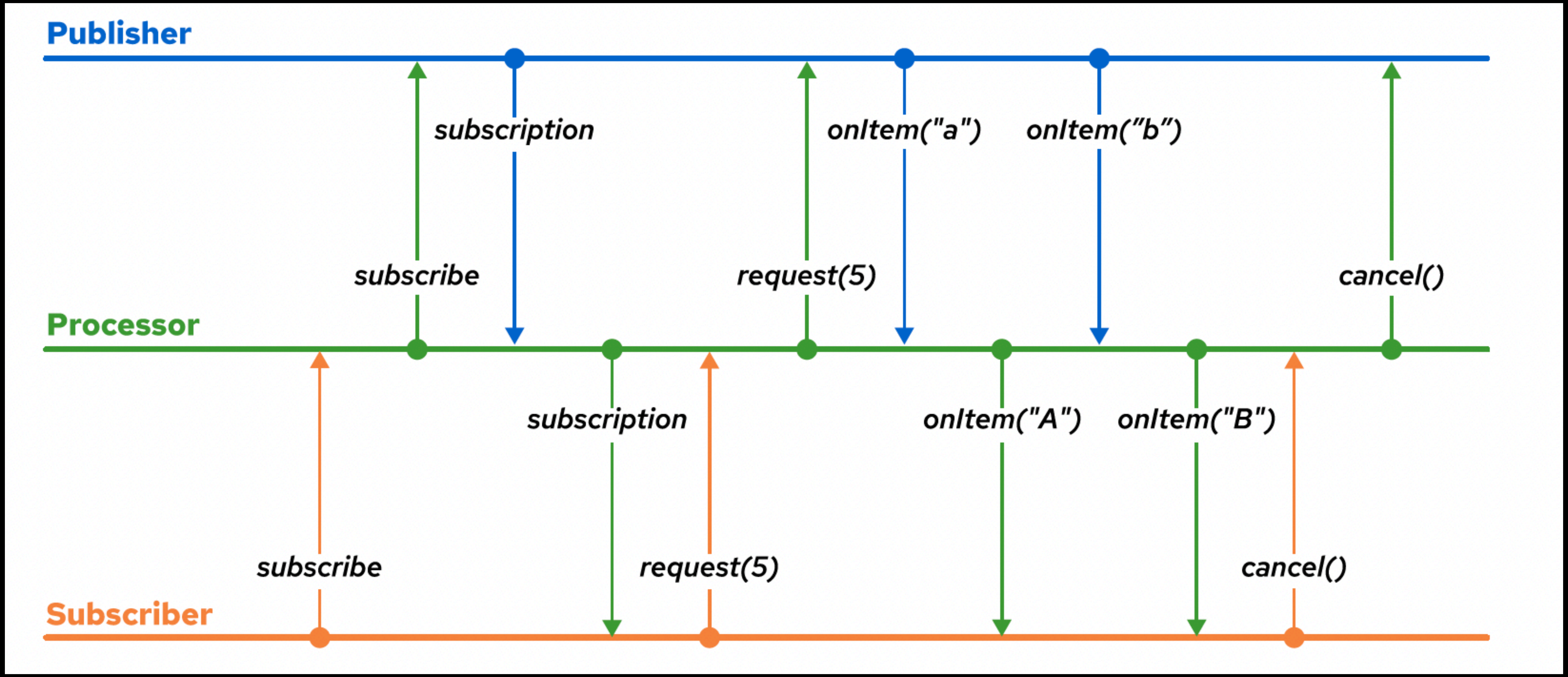


```
multi
    .onCompletion()
        .call( () -> file.close() );
```

Transformando Items

```
Multi.createFrom().items( tags )
    .subscribe()
        .with( item -> log( "Item proceesed: " + item ) );
    .onItem()
        .transform( tag -> tag.toLowerCase() )
    .onItem()
        .transform( lower -> "'" + lower + "'" )
```

Reactive workflows



Reactive workflows

```
Multi.createFrom().items( upstream )
    .onSubscription()
        .invoke( subscription -> log( "Upstream subscribed event" ) )
    .onRequest()
        .invoke( n -> log( "Downstream requested " + n + " items" ) )
    .onItem()
        .invoke( item -> log( "Item event: " + item ) )
    .onItem()
        .transformToUni( item -> externalCall( item ) )
    .subscribe()
        .with( item -> log( "Subscriber received " + item ) )
    .onFailure()
        .invoke( failure -> log( "Failed event: " + failure ) )
    .onCancellation()
        .invoke( () -> log( "Downstream cancelled event" ) )
    .onCompletion()
        .invoke( () -> log( "Completion event" ) )
```

Reactive panache

```
Panache.withTransaction( () -> entity.persist() );
```

Recursos

https://developers.redhat.com/promotions/building-reactive-microservices-in-java

https://smallrye.io/smallrye-mutiny/1.7.0/

https://hibernate.org/reactive/