actor

```
public fun <E> actor(
        context: CoroutineContext = DefaultDispatcher,
        … ,
        block: suspend ActorScope<E>.() → Unit
): SendChannel<E>
```

```
val squaresChannel: SendChannel<TableUpdate> = actor(UI) {
    for (update in channel)
        redrawTable(update)
}
```

select

```
val answer = select<String> {
    job.onJoin { "job finished" }
    asyncValue.onAwait { "completed with $it" }
    sendChannel.onSend("message") { "message sent" }
    receiveChannel.onReceive { "$it received" }
}
```