2-basic-example - simple-core-async

Benefits

In this exercise you'll gain understanding of the following:

- go blocks and how they do concurrency
- core.async queues
- go block timers

Assumptions

- You have Leiningen installed.
- You have an internet connection (if you don't have this – then we can copy the maven archive across)

Code to Read

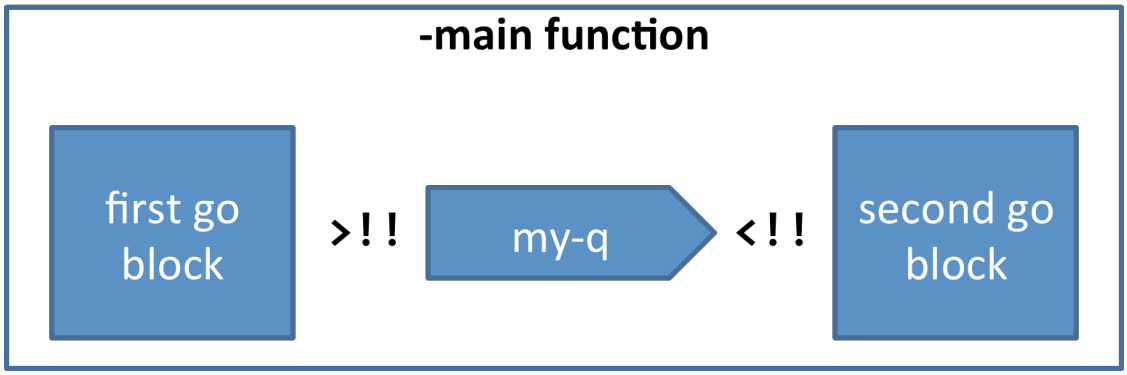
- `lambdajam-2014-core.async-workshop\2-basic-example\simple-core-async\src\simple_core_async\core.clj`

Things to Note In the Code

1. the two `go` blocks
2. the queue created with the `chan` function
3. the sequence of queue items in the `my-seq` symbol
4. items being added to the queue in the first `go` block
5. items being removed from the queue in the second `go` block
6. the `timeout` in the first `go` block
7. the absence of a `timeout` in the second `go` block

Code Model

This is a quick way to understand what is going on in the code:



Activities

1. Run the code with `lein run`
2. Change the `timeout` duration and run it again
3. Comment out the `timeout` and run it again (then put it back)
4. Put a `timeout` in the second `go`-block

5. Put a large `timeout` (2000) in the second `go` block, put a small `timeout` in the first `go` block (10), reduce the queue size to 1 in the `chan` function. Run it again.
6. Reset the code to the original, then comment out the `future` (and shutdown agents) and run it again.

Questions for Reflection

1. Why doesn't the second `go` block need a timeout?
2. Why do we need the future?
3. Do `go` blocks demonstrate concurrency or parallelism or both?
4. Could you build Conway's Game of Life using go blocks for each cell? If yes – how would you design it? If no, why not?
5. Some have called core.async 'syntax sugar over threads' – is that fair?