Channels

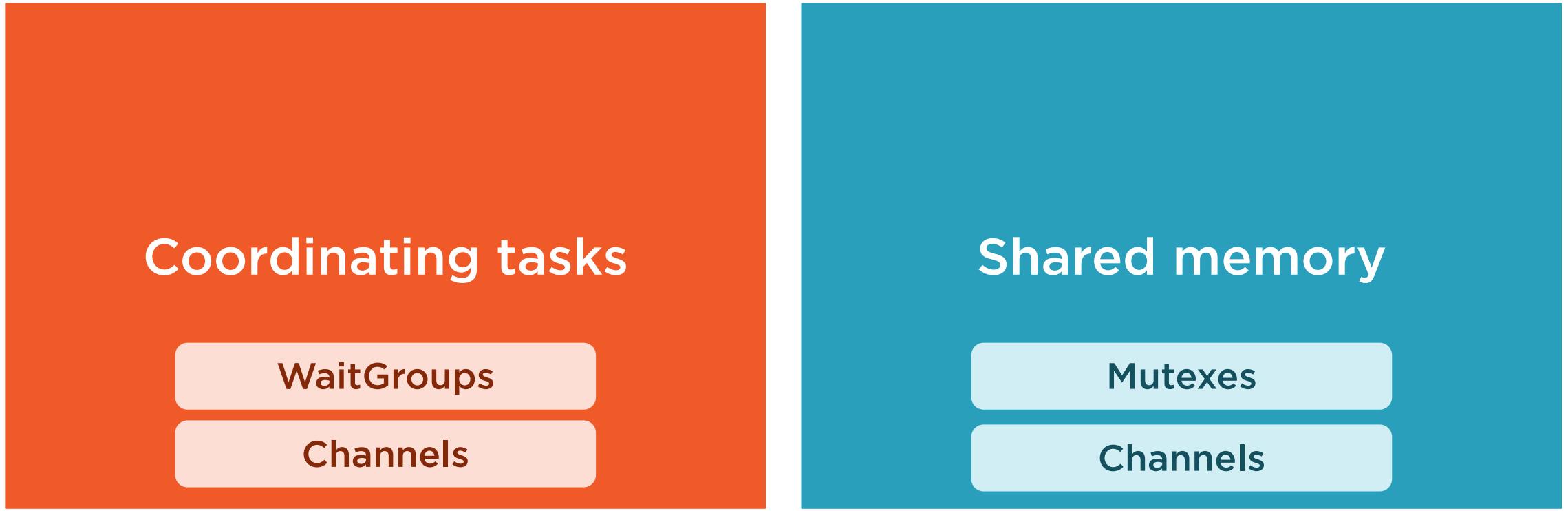
Michael Van Sickle

@vansimke

“Don’t communicate by sharing memory, share memory by communicating”

Rob Pike

Challenges with Concurrency



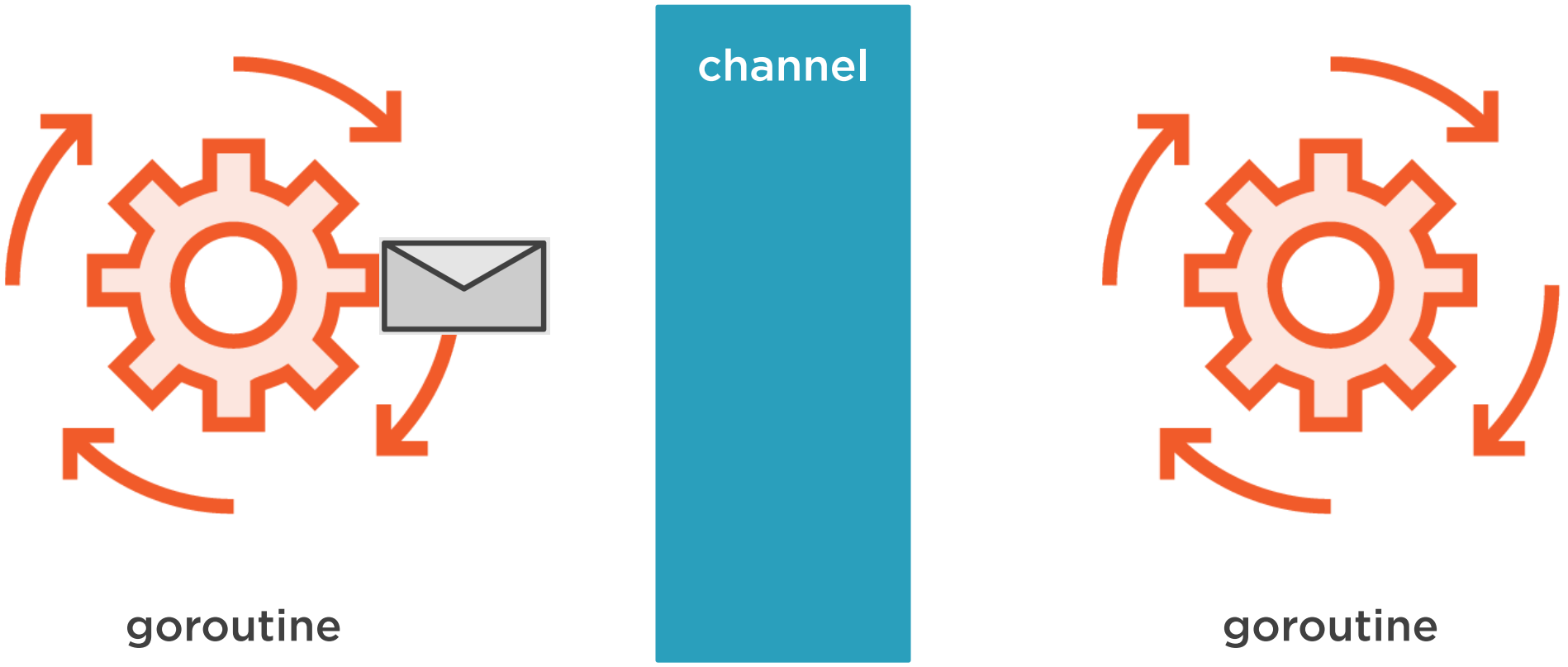
Channels

goroutine

mutexes
channels

goroutine

Channels



Overview

- Creating channels
- Buffered channels
- Channel types
- Closing channels
- Control flow

Creating Channels

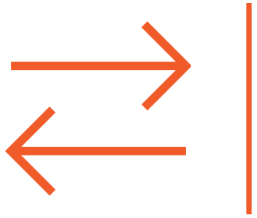
```
// create a channel

ch := make(chan int)


// create a buffered channel

ch := make(chan int, 5)
```

Channel Types



Bidirectional

Send-only

Receive-only

Channel Types

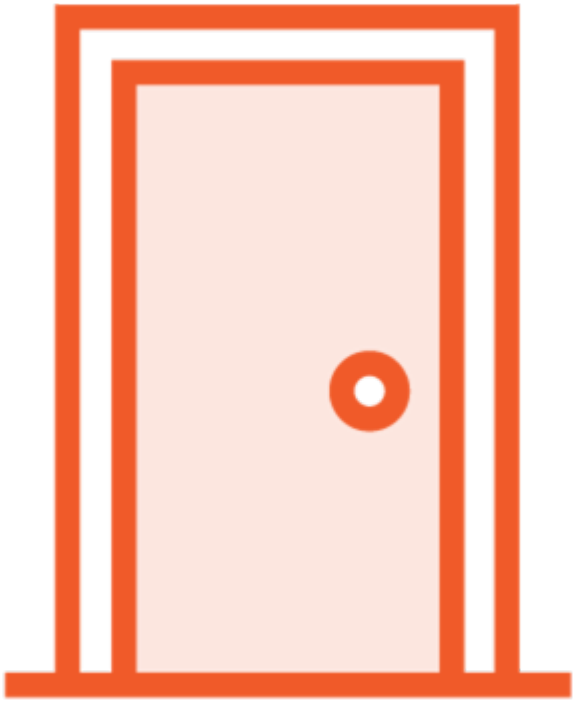
```
ch := make(chan int) // created channels are always bidirectional

func myFunction(ch chan int) { ... }          // bidirectional channel

func myFunction(ch chan<- int) { ... }        // send-only channel

func myFunction(ch <-chan int) { ... }        // receive-only channel
```

Closing Channels



Closed via the built-in close function

Cannot check for closed channel!

Sending new message triggers a panic

Receiving messages okay

- If buffered, all buffered messages available
- If unbuffered, or buffer empty, receive zero-value

Use comma okay syntax to check

Control Flow

If statements

For loops

Select statements

Select Statements

```
ch1 := make(chan int)

ch2 := make(chan string)

select {
        case i := <-ch1:

                ...

        case ch2 <- “hello”:

                ...

        default:

                // use default case for non-blocking select
}
```

Summary

Creating channels

Buffered channels

Channel types

- Bidirectional
- Send-only
- Receive-only

Closing channels

Control flow

- If
- For
- Select

Course Wrap-up

Goroutines

The sync package

Channels