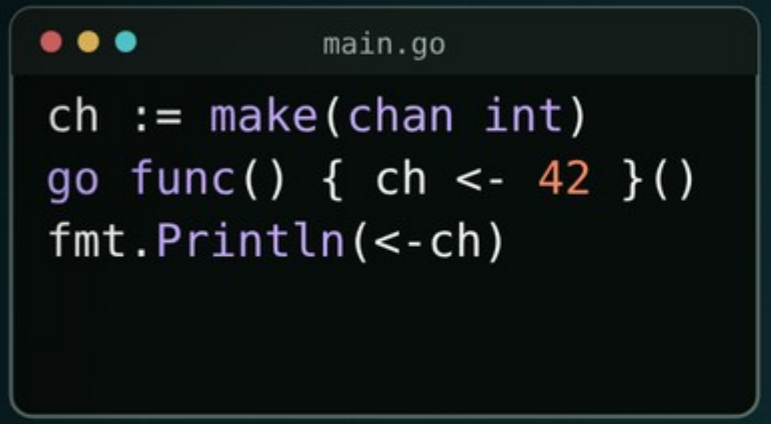
LESSON 9

# Go

## Goroutines






A FREE MYWEBUNIVERSITY COURSE

# GO PROGRAMMING

LESSON 9

## Goroutines and Channels

The go keyword, WaitGroup, channels, buffering, range and close

# Goroutines




- main() — starts
- go fetch(a) — goroutine 1
- go fetch(b) — goroutine 2

01

# Waiting with sync.WaitGroup



wait.go

```
var wg sync.WaitGroup
for _, url := range urls {
    wg.Add(1)
    go func() {
        defer wg.Done()
        fetch(url)
    }()
}
wg.Wait()                    // all done
fmt.Println("finished")
```

- When main returns, the program ends, even if goroutines are still running. A WaitGroup …
- Add one before starting each goroutine, call Done when it finishes, and Wait blocks until …

02

# Channels



channel.go

```
results := make(chan int)

go func() {
    results <- 6 * 7          // send
}()

answer := <-results           // receive (waits)
fmt.Println(answer)           // 42
```

- A channel is a typed pipe between goroutines. Send with the arrow pointing into the …
- An unbuffered channel makes the sender and receiver meet. Go's motto: do not communicate by …

03

# Buffered Channels, close and range



pipeline.go

```
jobs := make(chan int, 3)
go func() {
    for i := 1; i <= 5; i++ { jobs <- i }
    close(jobs)
}()
for j := range jobs {
    fmt.Println("job", j)
}
```

- make chan int comma three creates a buffered channel that holds three values before the …
- The sender calls close when done, and the receiver can range over the channel until it is …

04

# A Worker Pool



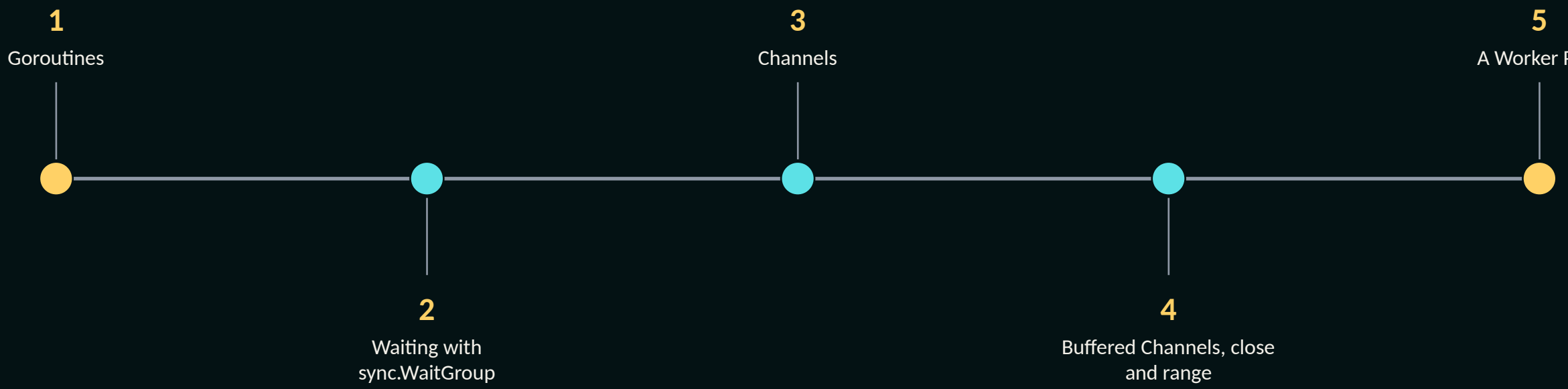
pool.go

```
func worker(id int, jobs <-chan string, out chan<- int) {
    for f := range jobs {
        out <- countLines(f)
    }
}
for w := 1; w <= 4; w++ {
    go worker(w, jobs, out)      // 4 workers
}
```

- Combine everything: a jobs channel, a results channel, and a fixed number of worker …
- This pattern controls how much work runs at the same time, perfect for downloading or …

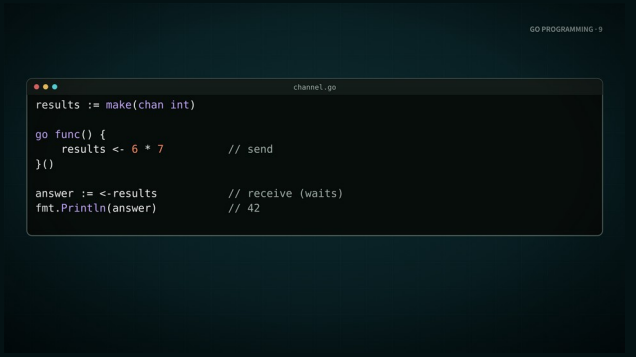
05

# Lesson Map


1
Goroutines
2
Waiting with sync.WaitGroup
3
Channels
4
Buffered Channels, close and range
5
A Worker Pool

# Sources and Further Reading



Thank you for watching

# MyWebUniversity.com

*Free technical education since 2000 — English and Dari*

@MyWebUniversity · @MyWebUniversity-Dari