Concurrency

Saber Mesgari

Golang Concurrency

- Ability of a program to do multiple things at the same time

- Concurrency in Golang is the ability for functions to run independent of each other

  - GoRoutines
  - Channels

Gorouting

- A goroutine is a lightweight thread managed by the Go runtime.

```
func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(100 * time.Millisecond)
        fmt.Println(s)
    }
}
func main() {
    go say("world")
    say("hello")
}
```

Channels

- Channels are a typed conduit through which you can send and receive values with the channel operator

```
ch := make(chan int)

ch <- v    // Send v to channel ch.
v := <-ch  // Receive from ch, and
           // assign value to v.
```

- In Channels Send and Receieve are blocking

Buffered Channels

- Channels can be buffered.

```
ch := make(chan int, 2)
ch <- 1
ch <- 2
fmt.Println(<-ch)
fmt.Println(<-ch)
```

Ranges And Close

- A sender can close a channel to indicate that no more values will be sent.

```
v, ok := <-ch
```

Select

- The `select` statement lets a goroutine wait on multiple communication operations.

- A select blocks until one of its cases can run, then it executes that case. It chooses one at random if multiple are ready.

Default Selection

- The default case in a select is run if no other case is ready.

- Use a default case to try a send or receive without blocking:

Mutex

- The default case in a select is run if no other case is ready.

- Use a default case to try a send or receive without blocking:

Two Column Slide

Just in case you need it:

- Multiple columns
- Huzzah