More On Go / What is Special about Go

“It is not the strongest of the species that survives, nor the most intelligent. It is the one that is the most adaptable to change.” Charles Darwin

1st. the news

1. Land registry in Mexico is moving onto blockchain.
2. FED Secretary, Not a fan of bitcoin: https://markets.businessinsider.com/currencies/news/bitcoin-price-cryptocurrency-should-be-curtailed-terrorism-concerns-yellen-2021-1-1029985692
3. Double Spend! https://markets.businessinsider.com/currencies/news/bitcoin-price-double-spend-flaw-critical-report-suggests-2021-1-1029990921

2nd. Purpose of a business

To make a profit for the owners of the business.

What is “Fiduciary Responsibility”. It means that you have been placed / are in a position of legal responsibility for managing somebody else’s money.

Go - What is Special

Compile Speed

It is really fast.

Garbage Collector

It is really fast and incremental.

Maps

Go has dictionary/maps

```
var m1 map[string]int
m1 = make(map[string]int)
m1["abc"] = 12
k := m1["abc"]
k2 := m1["xyz"]
```

```
k3, ok_t := m1["abc"]
k4, ok_f := m1["xyz"]
```

Observations 1. memory is not allocated to a map when it is declared. 2. You can just use `make` and `:=` to declare a map. 3. You can test to see if you have an un-allocated map by comparing to `nil`. 4. You can find out if a value is in a map.

Slices (Arrays)

An Array

```
var a1 [4]int
```

A slice

```
var s1 []int
```

What is a slice?

Allocating memory to a slice. Slices start out as “empty” or `nil`.

```
s1 = make ( []int, 5 )
s1 = make ( []int, 3, 6 )
```

Slice of slice:

```
s1 = s[1:2]
```

All of a slice or an array (how to convert an array to a slice)

```
s2 := s1[:]
```

Pitfalls!

Strings

Strings are immutable! Hot to denote a string.

Maps

A map is `var Name map[HashKeyType]ElementType`

Declare:

```
var Hw map[string]int
func init() {
        Hw = make( map[string]int )
}
```

or

```
Hw := make( map[string]int )
```

```
Hw["I80"] = 1421
Hw["US287"] = 841
```

Pull out the value and if a value is set.

```
vv := Hw["aaa"]
ww := Hw["I80"]
mm, found := Hw["I80"]
_, found2 := Hw["I90"]
```

Go Routines

Just call a function and have it run concurrency:

```
go QrDispatch()
```

Or put a function inline and have it run concurrently:

```
// ticker on channel - send once a minute
go func(n int) {
        for {
                time.Sleep(time.Duration(n) * time.Second)
                timeout <- "timeout"
                n_ticks++
        }
}(gCfg.TickerSeconds)
```

Note that the 2nd example is a “closure” where the value passed is then part of the function.

Note the `<-` operator is a inter-process communication operator (channel) built right into the language.

Note the `time.Sleep()` call. This sleep is on the go-routine, not the main code.

Concurrency Control

With concurrency comes concurrency control! This is what JavaScript(node.js) is missing.

Also the Go model is much more robust than other languages like Rust.

Also this is better than languages like Python that just don’t have any!

Declare a synchronization lock.

```
import "sync"
...
var aLock sync.Mutex
...

aLock.Lock()
aLock.Unlock()
```