

Go lang

system lang
web dev lang

- low-level programming language
- built by Google to solve Google types of problems
- high-frequency trading and really want quickly but don't want to write tediously like C.
- Go is like static type like ++ unlike python (คอมไพเลอร์ interpret ว่า what type is certain variables)
 & it gives you some speed.

- import "fmt" basic format
- ไม่ต้องการ semi colon ;
- use go run <filename> to run on console
- math.Sqrt(4)
↳ 1st letter is capitalized → that function will be exported by go
↳ if it's not capitalized → it is internal thing
- import ("math/rand")
 ↳ sub package of
 ↳ ไม่ต้องการ import แล้ว math/rand, math.Sqrt error เพราะเราต้อง import แล้ว "rand" ไม่ใช้ import แล้ว math. So, "math" "math/rand" (✓)
- เปิด Go document ได้โดย
(X) > godoc fmt.Println

- func add(x float64, y float64) float64 { }
↳ x, y คือ variable
↳ return คือ function()
- Declaration ▶ var num1 float64 = 5.6
- var num1, num2 float64 = 5.6, 9.5
- ประกาศใช้ num1, num2 outside ได้แบบ public static
↳ num1, num2 float64 := 5.6, 9.5
- for constant variable, const x int = 5
- func multiple(a, b string) (string, string) return a, b { }
- Convert ▶ var a int = 62
var b float64 = float64(a)

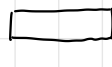
- x := a → x มี type int
- Pointer ▶ explanation in code file
- http.HandleFunc("/", index_handler)
homepage function to handle index page
ex. google.com/

- Struct ใช้เพื่อสร้าง constructor ใน class แต่สั้นกว่า

```
type car struct {  
    gas uint16  
    brake uint16  
    top-speed float64  
}
```

ชื่อ class
↳ use this name

```
func main() {  
    a_car := car { gas: 10, brake: 5,  
    ↳ a top-speed: 12.5 }  
    b_car := car { 10, 5, 12.5 } ก็ใช้แบบนี้แหละ better  
    fmt.Println(a_car.gas)  
    ↳ dot notation
```

- Methods in class that use variables in that struct to do some calculation / operation
func (c car) kmh() float64 {
 return c. 
 ↳ gas-pedal top-speed
}
in main() ...
fmt.Println(a_car.kmh())
- Pointer value : use the function to change variable's value in the struct


```
func (c *car) new-top-speed(newspeed float64) {  
    c.top-speed-kmh = newspeed  
    ↳ ไม่ต้องการ return แล้ว set ค่าใหม่ (อย่าใส่ใน return)  
}   
in main() ...  
fmt.Println(a_car.kmh())  
a_car.new-top-speed(500)  
fmt.Println(a_car.kmh()) // w/ new value  
* ที่ไม่ใส่ return  
c.top-speed-kmh = 500 ใช่มั้ย?  
func kmh() แล้วล่ะ?
```

- cons ▶ ถ้า func อันนี้ใช้ตัวแปรใน struct, แล้ว speed เปลี่ยนที่
- Pro ▶ It was permanently set because we use pointer

But we can convert this func to be pointer receiver instead of value receiver by changing to be c *car then the top speed will be modified (=500) and if we call other methods after call this method the new value will be used there

But แล้วทำไมไม่ใช้ pointer แทน value ไปได้?

Pointer receiver = สร้างตัวแปรใหม่เก็บ address เช่น c *car

คือ  เพื่อที่จะ modify ตัวแปรเดิมเสมอ
↳ pointer : จะใช้ชื่อ, เปลี่ยนโดยใช้ตำแหน่ง memory address เดิม

Value receiver = copy value แล้วสร้างตัวแปรใหม่เพื่อเก็บค่า
↳ so, no side effect to memory address ของตัวแปรก่อนหน้า

เราจะใช้อันไหน? เพื่ออะไร?

ใช้ pointer when → ① large scale struct . no need to copy, ② สามารถ modify the receiver cheaper

ใช้ value when → ① concern about data race & concurrency safe, ② Optimizing garbage collector
 ↳ ที่เราแล้ว clear ก็
③ small structs / basic type → value r. is very cheap (unless it requires pointers). value r. is efficient & clear

ใช้ func แทน method ไปได้? ได้! (write over)

```
func new-top-speed(c car, speed float64) car {  
    c.top-speed-kmh = speed  
    return c  
}   
in main... a_car = new-top-speed(a_car, 500)
```

this func returns

- Web (more) → document in code file
 - ⊗ if we declare sth. and it is not used, the program will be error.
- Internet → document in code file
- Parse XML → ① import encoding/xml ② SitemapIndex struct
 - ③ Location struct ④ main ⑤ func string()
- * we can get rid of ③ & ④ by fix ② as below


```
type SitemapIndex struct {
    Locations []string `xml:"sitemap>loc"`
}
```
- Looping → for example


```
a:=3
for x:=5; a<25; x+=3 {
    fmt.Println("Do stuff ", x)
    a+=4
}
```
- Web app, Mapping, Sitemap map, HTML templates
 - document in code file

- Applying template that highlighted as
 - Step 1: create a page of news aggregator then create a struct for this page
 - ① Title ② News → which it is NewsMap type to retrieve news data
 - Step 2: use the same struct from sitemap map which are
 - Sitemapindex: ๖๖๖ xml (sitemap>loc>xml),
 - News: ๖๖๖ xml ๖๖๖ ๖๖๖ Titles, keywords, Locations ๖๖๖ url,
 - News Map: keyword, Location
 - Step 3: take main function ที่ loop+get Locations ของ xml แล้วเอา n ที่ป้อน xml มาจะได้อะไร
 - n.Titles → key ของ newsMap
 - n.Keywords } newsMap → value
 - n.Locations }
 - อันนี้ newsMap ที่ป้อน type NewsMap

Goroutine & Channel

```
func Hello(name string) string {
    result := "Hello"+name
    fmt.Println(result)
    return result
}
```

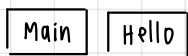
result is ...

start main
end main
Hello Beau

for example, in main

```
fmt.Println("start main")
name := "Beau"
go hello(name)
fmt.Println("end main")
time.Sleep(time.Second)
```

goroutine
ทำงานคู่กัน
2 ส่วนแยกกัน
แบบนี้



ไม่สามารถเข้าถึง
ผลของของ Hello ได้เลย
เลยให้ sleep ๑s Hello()

ดังนั้น, ใช้ channel ไป



```
fmt.Println("start main")
name := "Beau"
```

สร้าง channel
ชื่อ result โดย
ข้อมูลที่จะส่ง
เป็น string

```
result := make(chan string)
go hello(name)
fmt.Println("end main")
fmt.Println(<- result)
```

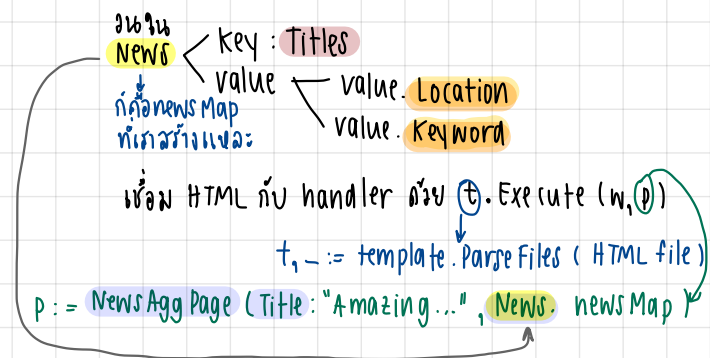
main รอจนกว่าทำงานผ่าน channel
result = no more sleep()

```
func Hello(name string, result chan<- string) {
    result := "Hello"+name
    fmt.Println(result)
    result <- output
}
```

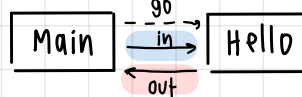
start main
end main
Hello Beau
Hello Beau

<-
channel
operator

Step 4: create HTML template ที่แสดง



สมมติว่าแบบ Bidirection (มี 2 channels)



← chan string

chan ← string

Goroutine Sync

Defer

Panic

I've described along with the code in gotut 9 go, gotut 10. go, gotut 11. go (in // comment)

Go Channels

check out gotut12.go for the code and detail as text.

Iterating channels

```
15 25Beaus-MacBook-Pro:softarch_go beau$ go run gotut12.go
0
10
5
30
25
15
20
40
45
35
fatal error: all goroutines are asleep - deadlock!
after
goroutine 1 [chan receive]:
main.main_for_iterating_channel()
/Users/beau/Desktop/softarch_go/gotut12.go:39 +0x10a
main.main()
/Users/beau/Desktop/softarch_go/gotut12.go:49 +0x20
exit status 2
```

we know what items are in fooVal, but we don't know when it's done. So, we add (close(fooVal)) after go foo's loop

but it returns nothing in terminal, why?

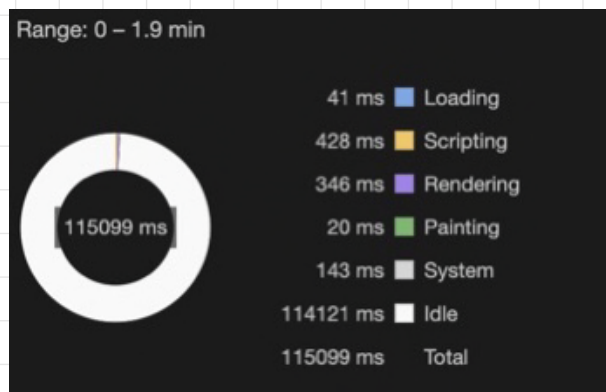
- All goroutines are off running and the program is finishing before they come back. So, we need the synchronize thing.
- After we add wg.Add(), wg.Done(), wait(), it still has error. Then we add **buffer** to fooVal := make(chan int, 10) because our channels block each other. but we know the amount of channels (10). So, we can control our flow.

* Note that **buffer** has to be \geq amount of channels
ex. For this case, buffer = 10, 50, ... \uparrow \checkmark
buffer = 5 \times

Web app Concurrency

From last webapp (gotut7.go), we inspect insight its performance. And it took around 1.9 mins (on my machine) to load. In our case, I was loading a massive table but the "idle" time was taking 115 seconds which is not good. So, we need some modifications

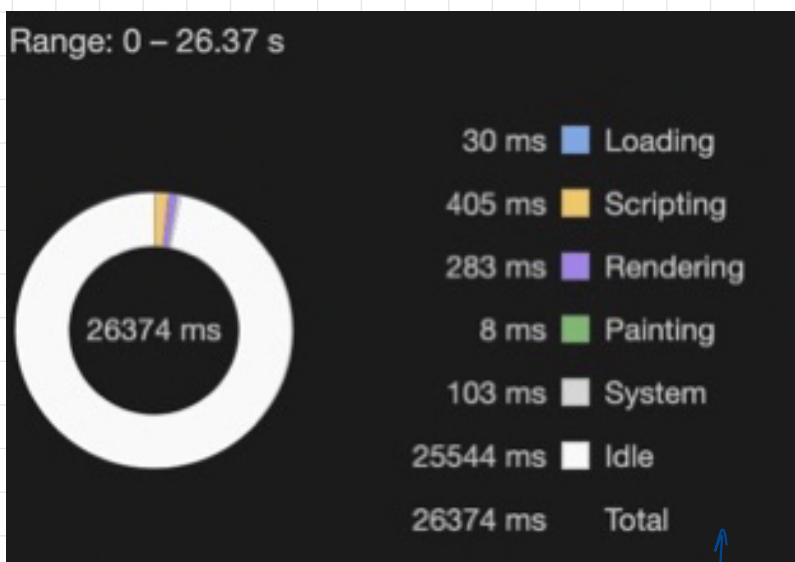
time where browser is just waiting for a response, "Golang's time"
every time we visit /Aaaa, it's repulling all of the sitemap



"How to modify with goroutines"

already in go doc inside the code file gotut13.go

and it went like this



* another way to improve the time is adding a real pagination faster than

ที่เรากำลังรับ
พร้อมกัน 1400 entries
จะมาเป็นตารางใหญ่
ใน REAL

แบ่ง response ของ API ออกเป็น page
เช่น (load more...)