Mastertech, AUG 16 2019

Basics Syntax #1

GolFz

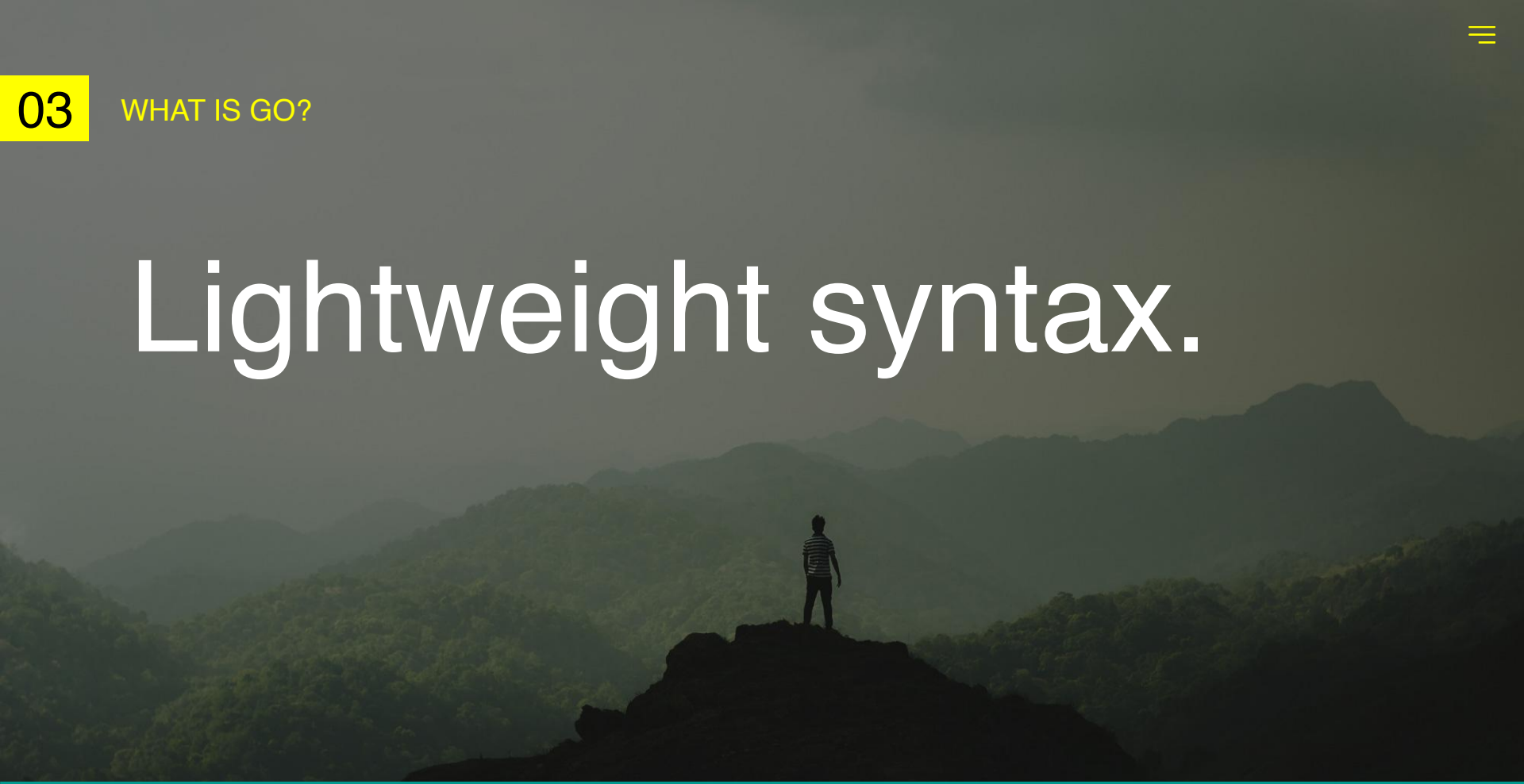
Mastertech

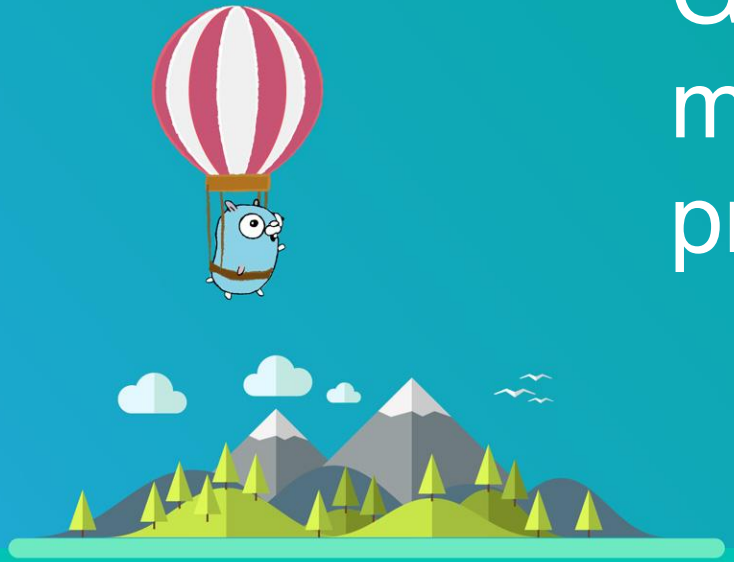
Engineer, R&D

Go is a modern, general purpose language.

Compiles to native machine code (32-bit and 64-bit x86, ARM).

Lightweight syntax.





“Go's purpose is to make its designers' programming lives better.”

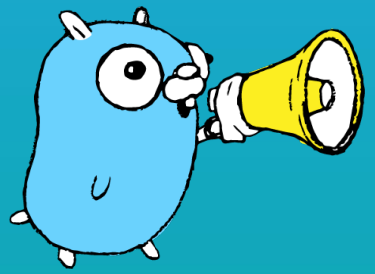
ROB PIKE

Gopher







Hi, Let’s talk about

THE GO PLAYGROUND

https://play.golang.org

Hello, ชาวโลก

```go
package main

import (
	"fmt"
)

func main() {
	fmt.Println("Hello, ชาวโลก")
}
```

https://play.golang.org/p/I8Nh6Bi5G1T

import

```go
package main

import "fmt"
import "math"

func main() {
	fmt.Println("result =", math.Sqrt(7))
}
```

https://play.golang.org/p/_JTkKPReqcW

เวลาใน Playground

```go
package main

import (
	"fmt"
	"math/rand"
)

func main() {
	fmt.Println("random :", rand.Intn(10))
}
```

https://play.golang.org/p/o5ElLOFGFql

01

Variables

Go’s basic variables

bool

string

int int8 int16 int32 int64
uint uint8 uint16 uint32 uint64

float32 float64

https://golang.org/pkg/go/types/#pkg-variables

Declare variables

```
package main

import (
	"fmt"
)

func main() {
	var a, b, c bool
	fmt.Println(a, b, c)
}
```

https://play.golang.org/p/IKywX3zvh3w

Declare variables (summary)

```go
package main

import (
        "fmt"
)

func main() {
        var a, b, c bool
        var d, e, f bool = true, false, true
        var g, h, i = false, 10, "hello"
        j, k, l := 14.5, false, "apple"
        fmt.Println(a, b, c, d, e, f, g, h, i, j, k, l)
}
```

https://play.golang.org/p/ymu8tgs-WT3

Go is strong type

```go
package main

import (
	"fmt"
)

func main() {
	var a int8 = 2
	var b int64 = 3
	fmt.Println("a + b =", a + b)
}
```

https://play.golang.org/p/SdjH5Hc9DUo

02

Looping (for)

For Loop

```go
package main

import (
	"fmt"
)

func main() {
	for i := 0; i < 10; i++ {
		fmt.Println(i)
	}
}
```

https://play.golang.org/p/xqcgSLnEjjy

While - Do Loop

```go
package main

import (
	"fmt"
)

func main() {
	i := 0
	for i < 10 {
		fmt.Println(i)
		i += 1
	}
}
```

https://play.golang.org/p/jUmykVDJkRo

Do - While Loop

```go
package main

import (
	"fmt"
)

func main() {
	i := 0
	for {
		fmt.Println(i)
		if i >= 10 {
			break
		}
		i += 1
	}
}
```

https://play.golang.org/p/wDg2Dmn7McT

Foreach (range) with Array

```go
package main

import (
	"fmt"
)

func main() {
	arr := [5]int{1, 2, 3, 4, 5}
	for index, element := range arr {
		fmt.Println(index, element)
	}
}
```

https://play.golang.org/p/8YSFIVy9lv8

Foreach (range) with Map

```go
package main

import (
	"fmt"
)

func main() {
	m := map[string]int{"foo": 1, "bar": 2}
	for key, value := range m {
		fmt.Println(key, value)
	}
}
```

https://play.golang.org/p/CPMbqgF0wY4

Foreach (range) with String

```go
package main

import (
	"fmt"
)

func main() {
	s := "Hello"
	for index, element := range s {
		fmt.Println(index, element)
	}
}
```

https://play.golang.org/p/jm1DZ1ecP49

03

Condition
(if / else)

IF

```go
package main

import (
	"fmt"
)

func main() {
	if 8%4 == 0 {
		fmt.Println("8 is divisible by 4")
	}
}
```

IF - ELSE

```go
package main

import (
        "fmt"
)

func main() {
        if 7%2 == 0 {
                fmt.Println("7 is even")
        } else {
                fmt.Println("7 is odd")
        }
}
```

IF - ELSE - IF

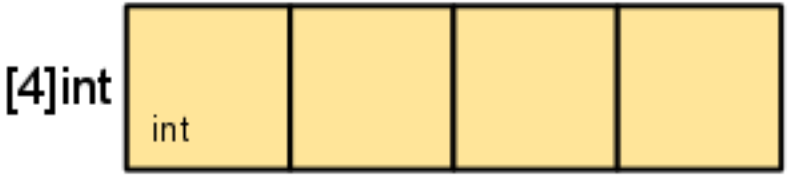
```go
package main

import (
	"fmt"
)

func main() {
	if num := 9; num < 0 {
		fmt.Println(num, "is negative")
	} else if num < 10 {
		fmt.Println(num, "has 1 digit")
	} else {
		fmt.Println(num, "has multiple digits")
	}
}
```

https://play.golang.org/p/T-imI1usaXt

04

Array

Basic of Array



Create an Array

```go
package main

import (
	"fmt"
)

func main() {
	var a [5]int
	fmt.Println("emp:", a)
}
```

https://play.golang.org/p/r7ljp2CeGRr

Set & Get a value with index

```go
package main

import (
        "fmt"
)

func main() {
        var a [5]int
        a[4] = 100
        fmt.Println("arr:", a)
        fmt.Println("get:", a[4])
}
```

https://play.golang.org/p/uwbhRdD2P6V

Declare & Initialize an Array

```go
package main

import (
	"fmt"
)

func main() {
	a := [5]int{1, 2, 3, 4, 5}
	fmt.Println("arr:", a)
}
```

https://play.golang.org/p/v-Hy0d3Tazl

Let’s compiler count the array elements for you

```go
package main

import (
	"fmt"
)

func main() {
	a := [...]int{1, 2, 3, 4, 5}
	fmt.Println("arr:", a)
}
```

https://play.golang.org/p/lrLCw0R2_wU

Length of an Array

```go
package main

import (
	"fmt"
)

func main() {
	a := [...]int{2, 4, 6, 8, 10, 12}
	fmt.Println("len:", len(a))
}
```

https://play.golang.org/p/8XzKn8JLrHq

Multi-Dimensional Array (1)

```go
package main

import (
        "fmt"
)

func main() {
        var twoD [2][3]int
        for i := 0; i < 2; i++ {
                for j := 0; j < 3; j++ {
                        twoD[i][j] = i + j
                }
        }
        fmt.Println("2d: ", twoD)
}
```

https://play.golang.org/p/HKWIuue_m4d

Multi-Dimensional Array (2)

```go
package main

import (
        "fmt"
)

func main() {
        var twoD = [2][3]int{
                {0, 1, 2},
                {3, 4, 5},
        }
        fmt.Println("2d: ", twoD)
}
```

https://play.golang.org/p/_aHXb7GATH5

Foreach (range) with Array

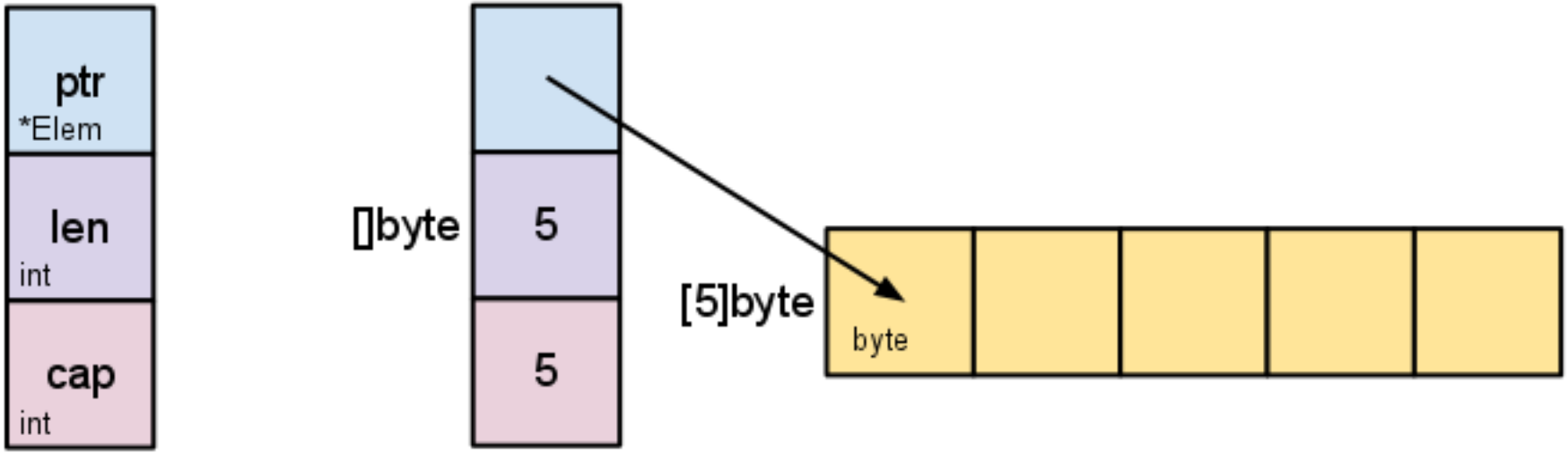
```go
package main

import (
	"fmt"
)

func main() {
	arr := [5]int{1, 2, 3, 4, 5}
	for index, element := range arr {
		fmt.Println(index, element)
	}
}
```

https://play.golang.org/p/8YSFIVy9lv8

05

Slice

Basic of Slice



Create a Slice

```go
package main

import (
	"fmt"
)

func main() {
	s := make([]int, 3)
	fmt.Println("emp:", s)
}
```

https://play.golang.org/p/y2-IQsBcWFz

Set & Get a value

```go
package main

import (
        "fmt"
)

func main() {
        s := make([]int, 3)
        s[1] = 5
        fmt.Println("slice:", s)
        fmt.Println("get:", s[1])
}
```

https://play.golang.org/p/atXRy1SFoIW

Append a new value

```go
package main

import (
	"fmt"
)

func main() {
	s := make([]int, 3)
	s = append(s, 5)
	fmt.Println("slice:", s)
}
```

https://play.golang.org/p/dvkCKFDA6FZ

Create a empty Slice

```go
package main

import (
	"fmt"
)

func main() {
	s := []int{}
	fmt.Println("slice:", s)
}
```

https://play.golang.org/p/OtX2N6Tdsy2

Declare & Initialize a Slice

```go
package main

import (
	"fmt"
)

func main() {
	s := []int{1, 2, 3, 4, 5}
	fmt.Println("slice:", s)
}
```

https://play.golang.org/p/P8d-ipBLoAE

Length of a Slice

```go
package main

import (
	"fmt"
)

func main() {
	s := []int{1, 2, 3, 4, 5}
	fmt.Println("len:", len(s))
}
```

https://play.golang.org/p/vkrNZGxfpc7

Copy a Slice

```go
package main

import (
	"fmt"
)

func main() {
	s := []int{1, 2, 3, 4, 5}
	c := make([]int, len(s))
	copy(c, s)
	fmt.Println("copy:", c)
}
```

https://play.golang.org/p/ivyWdUWSRmu

Slice operator

```go
package main

import (
        "fmt"
)

func main() {
        s := []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
        fmt.Println("s: ", s)
        fmt.Println("s1:", s[:])
        fmt.Println("s2:", s[2:])
        fmt.Println("s3:", s[:5])
        fmt.Println("s4:", s[2:5])
}
```

https://play.golang.org/p/wqLI0tLdmVU

ใช้บ่อย ๆ

```go
package main

import (
	"fmt"
)

func main() {
	s := []int{}
	s = append(s, 1)
	s = append(s, 2)
	s = append(s, 3)
	fmt.Println("slice:", s)
}
```

https://play.golang.org/p/3YoIDTE5DkB

Foreach (range) with Slice

```go
package main

import (
	"fmt"
)

func main() {
	s := []int{2, 4, 6, 8, 10}
	for index, element := range s {
		fmt.Println(index, element)
	}
}
```

https://play.golang.org/p/tNc_Id8Yfg4

06

Map

Create a Map (1)

```go
package main

import (
	"fmt"
)

func main() {
	m := make(map[string]int)
	fmt.Println("map:", m)
}
```

https://play.golang.org/p/btTHiBDglgm

Create a Map (2)

```go
package main

import (
	"fmt"
)

func main() {
	m := map[string]int{}
	fmt.Println("map:", m)
}
```

https://play.golang.org/p/av--EU8c5RU

Set & Get Key / Value

```go
package main

import (
        "fmt"
)

func main() {
        m := make(map[string]int)
        m["one"] = 1
        m["two"] = 2
        fmt.Println("value:", m["two"])
}
```

https://play.golang.org/p/pHBrgevmV81

Optional second return value

```go
package main

import (
        "fmt"
)

func main() {
        m := make(map[string]int)
        m["one"] = 1
        m["two"] = 2
        v1, v2 := m["one"]
        fmt.Println("v1:", v1, "   v2:", v2)
}
```

https://play.golang.org/p/jE5ffXeEpFJ

Length of a Map

```go
package main

import (
	"fmt"
)

func main() {
	m := make(map[string]int)
	m["one"] = 1
	m["two"] = 2
	fmt.Println("len:", len(m))
}
```

https://play.golang.org/p/E1FE45Fqufl

Remove key / value from a Map

```go
package main

import (
        "fmt"
)

func main() {
        m := make(map[string]int)
        m["one"] = 1
        m["two"] = 2
        fmt.Println("map:", m)
        delete(m, "two")
        fmt.Println("map:", m)
}
```

https://play.golang.org/p/vXEx7Xbb4kO

Foreach (range) with Map

```go
package main

import (
	"fmt"
)

func main() {
	m := map[string]int{"foo": 1, "bar": 2}
	for key, value := range m {
		fmt.Println(key, value)
	}
}
```

https://play.golang.org/p/CPMbqgF0wY4

07

Function

Basic

```go
package main

import "fmt"

func hello() {
	fmt.Println("สวัสดีชาวโลก")
}

func main() {
	hello()
}
```

https://play.golang.org/p/AjPc4eoTeHW

Function with parameter(s)

```go
package main

import "fmt"

func plus(a int, b int) {
	fmt.Println("plus :", a+b)
}

func main() {
	plus(1, 2)
}
```

https://play.golang.org/p/lXldnkaENfY

Variadic Functions (Function with infinite parameters)

```go
package main

import "fmt"

func show(a ...int) {
        for i, v := range a {
                fmt.Println("index:", i, "value:", v)
        }
}

func main() {
        show(1, 2, 3)
}
```

https://play.golang.org/p/zgHMx92TSOI

Return Values

```go
package main

import "fmt"

func plus(a int, b int) int {
	return a + b
}

func main() {
	res := plus(2, 3)
	fmt.Println("res:", res)
}
```

https://play.golang.org/p/faIwUaV9BSw

Multiple Return Values

```go
package main

import "fmt"

func getPosition() (int, int) {
        return 3, 7
}

func main() {
        x, y := getPosition()
        fmt.Println("x:", x, "y:", y)
}
```

https://play.golang.org/p/JfqVB5OiT5W

Return Error

```go
import "errors"

func dict(w string) (string, error) {
	m := map[string]string{"one":"หนึ่ง", "two":"สอง"}
	if v, hasWord := m[w]; hasWord {
		return v, nil
	}
	return "", errors.New("No this word")
}

func main() {
	s, err := dict("two")
	if err != nil {
		fmt.Println(err.Error())
		return
	}
	fmt.Println("s:", s)
}
```

https://play.golang.org/p/6txlUzSlSQu

Closures (1)

```go
package main

import "fmt"

func plus() func(a int, b int) int {
        return func(a int, b int) int {
                return a + b
        }
}

func main() {
        fn := plus()
        fmt.Println("plus:", fn(2, 3))
}
```

https://play.golang.org/p/e4z38ZibQCJ

Closures (2)

```go
package main

import "fmt"

func main() {
	fn := func(a int, b int) int {
		return a + b
	}
	fmt.Println("plus:", fn(2, 3))
}
```

https://play.golang.org/p/-T2Ltx92j2q

08

Defer

Defer

```go
package main

import "fmt"

func main() {
	defer fmt.Println("World")

	fmt.Println("Hello")
}
```

https://play.golang.org/p/pqqQzUDy1ma

Order of Defers

```go
package main

import "fmt"

func main() {
	defer fmt.Println("1")
	defer fmt.Println("2")

	fmt.Println("3")
}
```

https://play.golang.org/p/h1KDAUMI4rn

Q & A