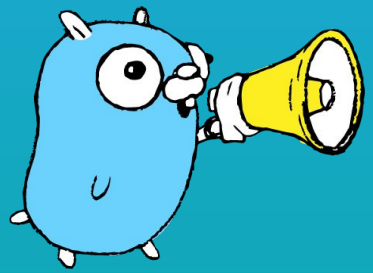
FOR FREECODECAMP TORONTO

INTRO TO GO

Alex Kallaway

Dev

@ka11away



Go Gopher Go!

SECTION ONE

About Go

Where Created & Who's Behind Go

- Created at Google
- Released as Open Source Software in 2009
- Creators: Robert Griesemer, Rob Pike, and Ken Thompson
- They also worked on (created/contributed to) C, B, Unix, Unicode, JVM, and others

Go's Values

- Thoughtful
- Simple
- Efficient
- Reliable
- Productive
- Friendly

Go's Main Features

- Go is a modern, general purpose language
- Strongly typed, compiled
- Automatic garbage collection
- Concurrency as a core language feature
- Unique approach to error handling
- No classes, structs and interfaces instead
- Simple, consistent language design

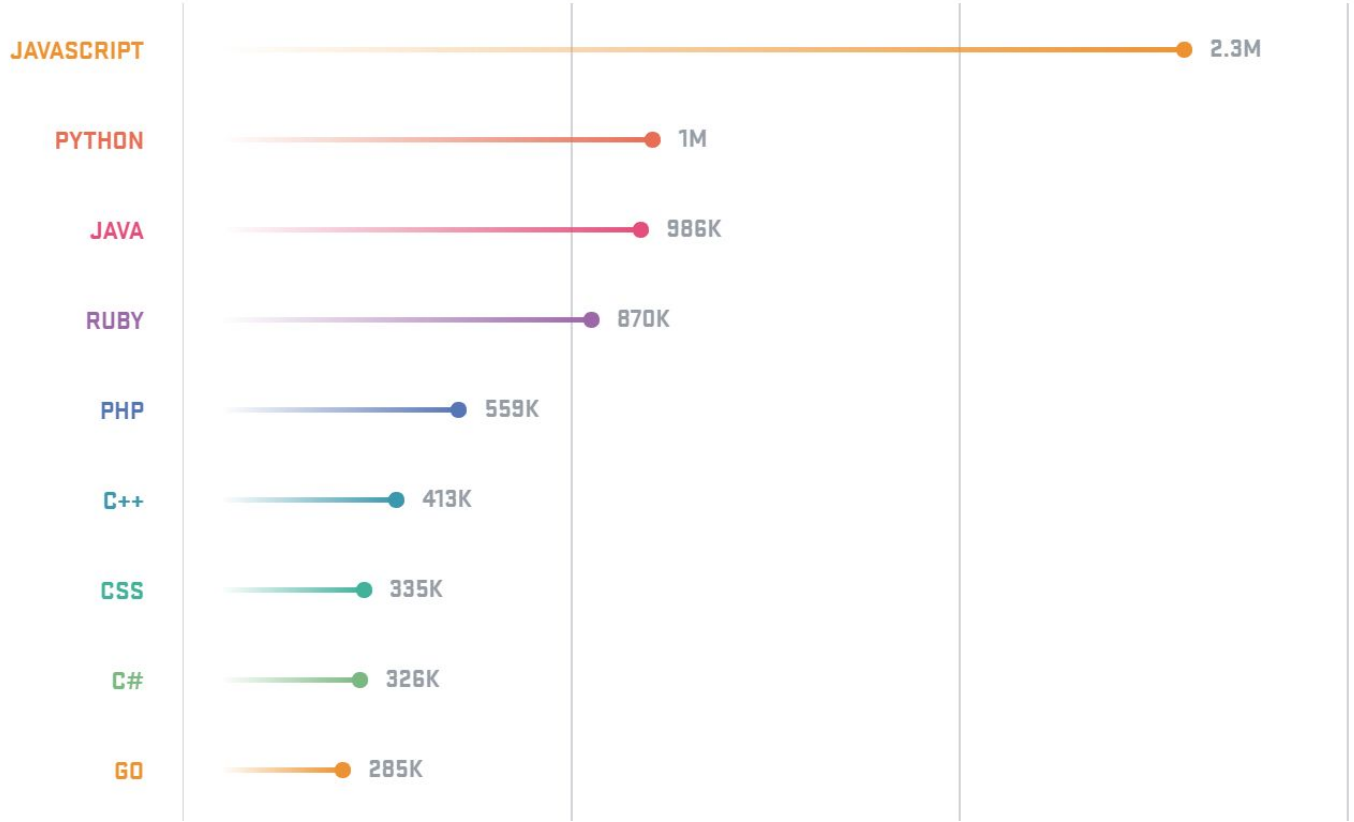
SECTION TWO

Why Learn Go?

Why Go?


Interest over time
100
75
50
25
Oct 2013
golang 17
Note
Oct 1, 2009
Jul 1, 2011
Apr 1, 2013
Jan 1, 2015
Oct 1, 2016

9th out of 15 Most Popular Languages on Github (by Pull Requests)

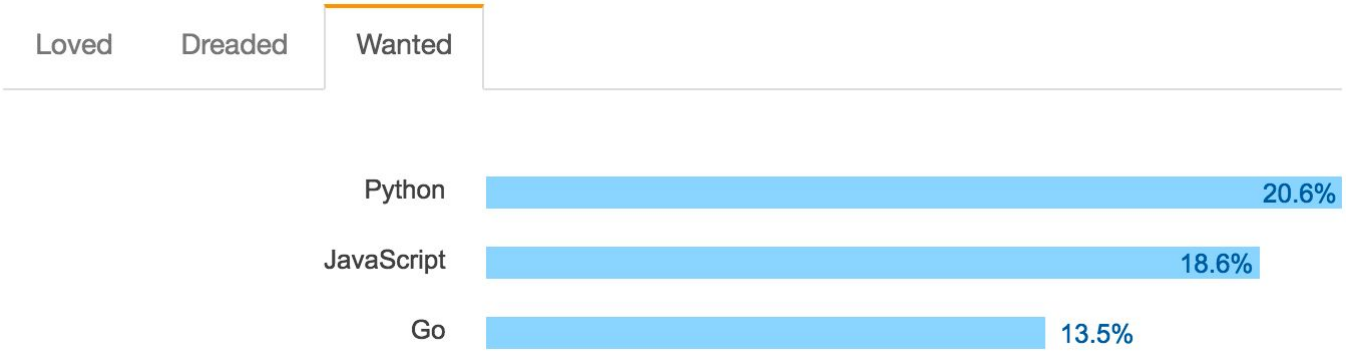


Why Go?

Most Loved, Dreaded, and Wanted Languages

Loved | Dreaded | Wanted

Why Go?



Companies/Projects Using Go

- Google
- YouTube
- Apple
- Dropbox
- Docker
- BBC
- The Economist
- The New York Times
- IBM
- Twitter
- Facebook

Why Learn Go?

- It’s insanely fast
- Concurrency features built into the core
- Rich standard library (+ Networking packages are available in standard library)
- Simple, consistent design (think Unix)
- Moore’s Law is failing

Why Learn Go?

- Developers love it - it’s pleasant to write code in it
- It’s rapidly gaining popularity
- Supports Unicode by default
- Incredible quality of documentation

Go is Cross-Platform

- It is used for different platforms, including Windows, Linux, Unix and BSD versions and mobile devices (starting from 2015). In addition, it compiles well on many OS's.

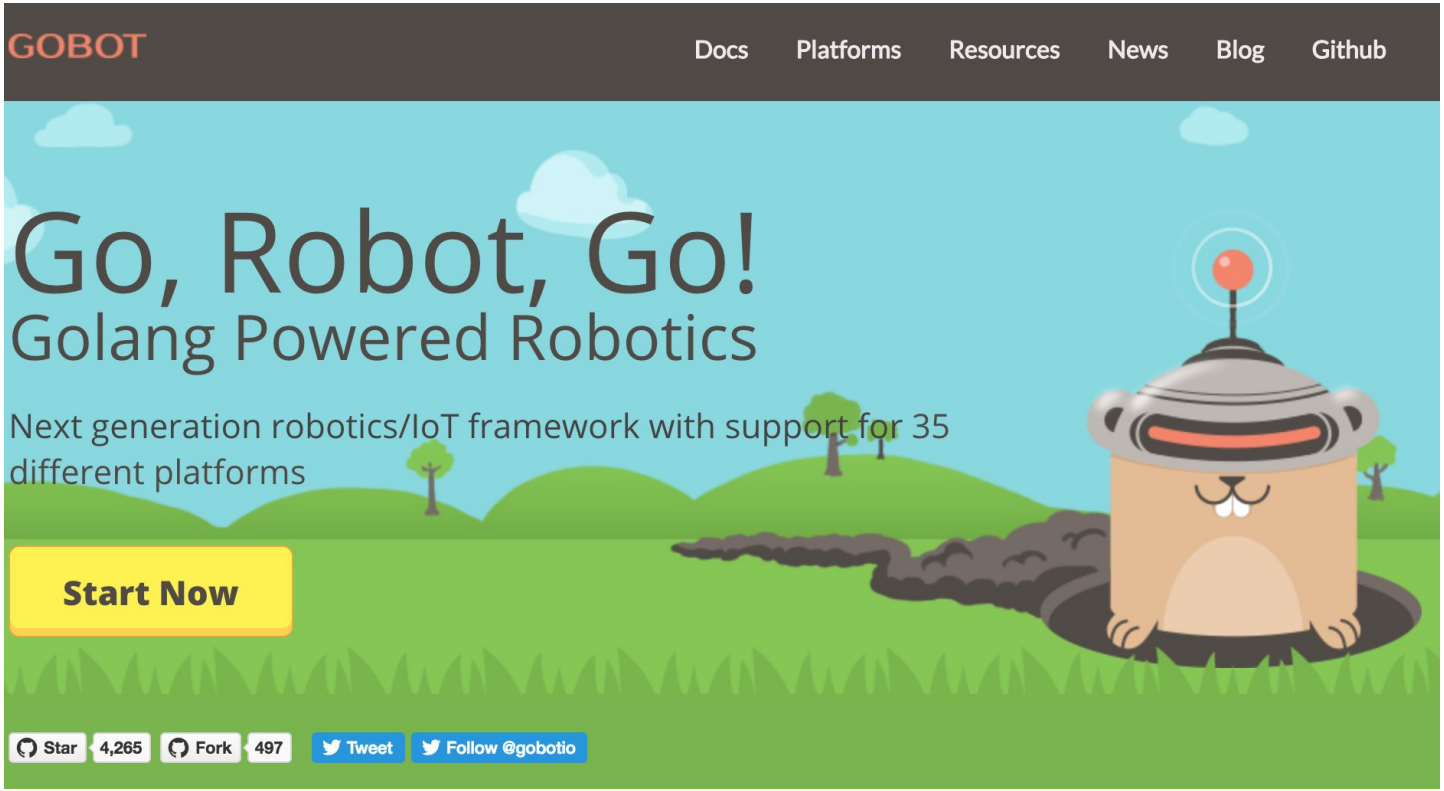
Why Learn Go?

- Go empowered the creation of new bold open source projects like Docker, Kubernetes and Ethereum
- Go's ability to handle large amounts of load with less memory and CPU cycles translates into savings in servers and hardware costs. There are stories of organizations going from 30 servers to just 2 handling the same load by migrating to Go.

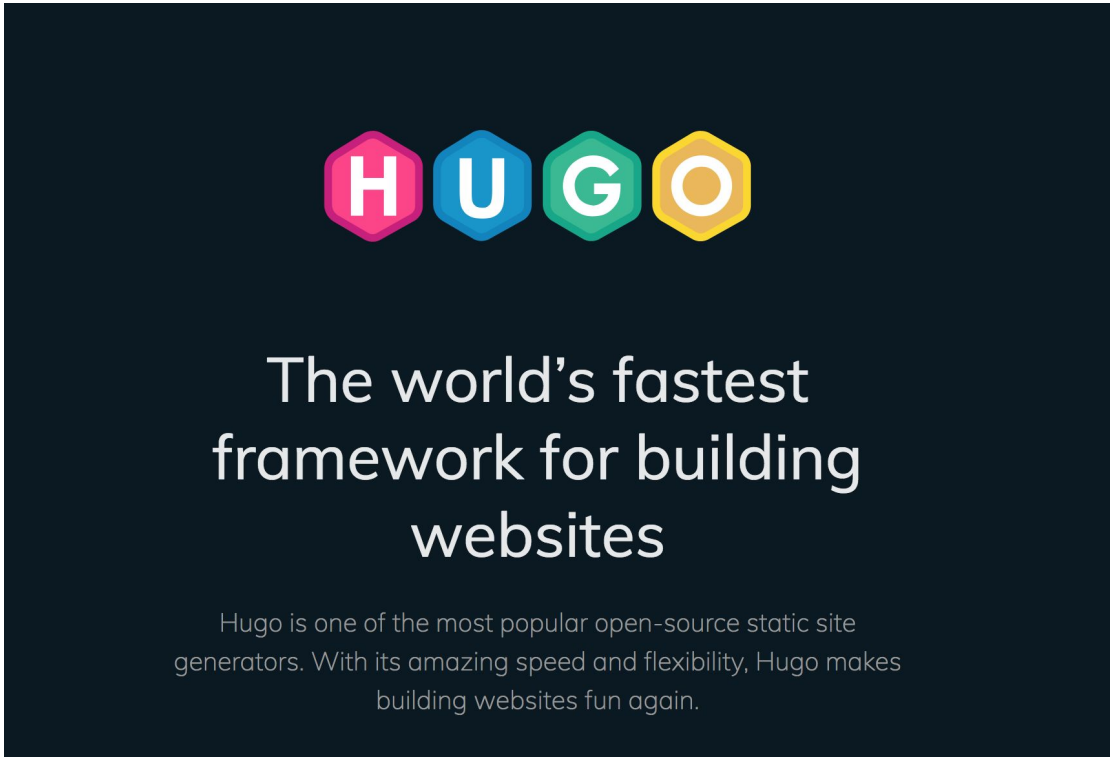
SECTION THREE

Projects to check out

Gobot.io


GOBOT
Docs
Platforms
Resources
News
Blog
Github
Go, Robot, Go!
Golang Powered Robotics
Next generation robotics/IoT framework with support for 35 different platforms
Start Now
Star 4,265
Fork 497
Tweet
Follow @gobotio

Hugo - Static Site Generator (gohugo.io)



SECTION FOUR

On Syntax

Syntax

- No semicolons
- If/Else statements and loops don’t use ()
- Everything should go through go fmt tool
- You can only use “” for strings, not single quotes - these are reserved for ‘runes’ (Unicode characters)
- Anything named with a Capital letter is exported

SECTION FIVE

Let’s get into code

HELLO WORLD

```
package main

import "fmt"

func main() {
    fmt.Println("hello world")
}
```

To run the program, put the code in hello-world.go and use go run.

VARIABLES I

```
var a = "initial"
fmt.Println(a)

var b, c int = 1, 2
fmt.Println(b, c)

var d = true
fmt.Println(d)
```

You can declare multiple variables at once. Go will infer the type of initialized variables.

VARIABLES II

```
var e int
fmt.Println(e)

f := "short"
fmt.Println(f)

const n = 500
```

Variables declared without a corresponding initialization are zero-valued. For example, the zero value for an int is 0.

LOOPS IN GO I

```
i := 1
for i <= 3 {
  fmt.Println(i)
  i = i + 1
}

for j := 7; j <= 9; j++ {
  fmt.Println(j)
}
```

for is Go's only looping construct.

LOOPS IN GO II

```
for {
  fmt.Println("loop")
  break
}
```

for {} is an infinite loop. We can ‘break’ or ‘continue’ within a loop

IF/ELSE STATEMENT

```
if a == 7 {
  // do something
}
```

There are no () brackets for the condition in an if else condition.

FUNCTIONS

```
func square(a int) int {
  return a * a
}

func sumThree(a, b, c int) int {
  return a + b + c
}
```

Go requires explicit returns, i.e. it won't automatically return the value of the last expression.

ARRAYS

```
var arr [3]int

arr[1] = 123

b := [5]int{1, 2, 3, 4, 5}

fmt.Println(len(b))
```

You can declare and initialize an array in one line.

SLICES

```
var arr []int

b := []int{“cat”,”dog”,”mouse”,}

append(b, “pigeon”)

c := make([]string, len(b))
copy(c, b)
```

Unlike arrays, slices are typed only by the elements they contain (not the number of elements).

MAPS I

```
people := map[string]int{
  “Bob”: 35, “Richard”: 23, “Kate: 28,
}
// get
people[“Bob”] // prints 35
// set
people[“Olivia”] = 27
// delete
delete(people, “Bob”)
```

Go provides a built-in map type that implements a hash table.

MAPS II

```
// Initializes a map with space for 15 items
m := make(map[string]int32, 15)

// check if the items exists
r1, ok := m["route"]
if ok {
 // do something with value
}
```

STRUCTS

```
type pet struct {
  name string
  kind string
  age int
}

myCat := pet{ “Dino”, “cat”, 3}
yourDog := pet{
  name: “Barky”
  kind: “dog”,
  age: 5,
}
```

HELPFUL STRUCTURES

```
a := []int{12, 34, 45}
for i, num := range a {
    fmt.Println(num)
}

func twoSquares(a, b int) (int, int) {
    return a * a, b * b
}
```

Range, Multiple returns

PRINT FUNCTIONS

```
fmt.Printf(“Hello, My name is %v”, “Slim Shady”)
```

There are multiple “print verbs” available

SECTION SIX

Get Set Up 

Get set up

1. Download Go: https://golang.org/dl
2. Install Go: https://golang.org/doc/install
3. Best Environments: a) VS Code + Go plugin b) JetBrains GoLand

SECTION SEVEN

Exercises 

Exercises

a. Reverse a string
b. Create a program that contains a function that converts temperature from Celsius to Fahrenheit
c. Create a type ‘person’, with name, age, and profession. Then create a slice of people (with info filled in). Print a sentence about each of the people in this structure:
“*Name* is *Age* years old. *Name* is a *Profession*.”
d. Create a program that on ‘go run main.go’ downloads a random image from this API:
https://picsum.photos/200/300/?random

SECTION EIGHT

Example Snippets

Examples

1. Interfaces example: https://github.com/kallaway/golang-snippets/blob/master/interfaces/shape/main.go
2. Save Images (error handling example): https://github.com/kallaway/golang-snippets/blob/master/scripts/save-image/main.go
3. Simple server: https://github.com/kallaway/golang-snippets/blob/master/web/001-hello-server/main.go

SECTION NINE

Next Steps

Your Next Steps...

1. Tour of Go https://tour.golang.org
2. Go by Example https://gobyexample.com
3. Effective Go https://golang.org/doc/effective_go.html
4. Exercism http://exercism.io/languages/go/about
5. Gophercises https://gophercises.com
6. Everything else: https://github.com/avelino/awesome-go

Bonus for those who stuck around:

a) https://github.com/ashleymcnamara/gophers
b) https://gopherize.me