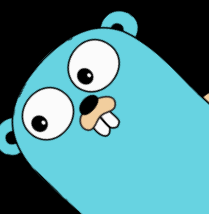


# An introduction to Go

Nils Nieuwenkamp  
13th of December, 2019



# Go?

- Programming language
- Developed by: Robert Griesemer, Rob Pike, Ken Thompson (Unix, UTF-8, B (predecessor C))

"Go is syntactically similar to C, but with:"

memory safety  
garbage collection  
structural typing  
CSP-style concurrency

# Do we really need more programming languages?

R,R,&K: YES

Me: But why?

R, R, & K: We need higher programming productivity in an era of multicore, networked machines and large codebases.

Me: If you say so.

R, R, & K: Also: we hate C++.

# Main design characteristics

## C-like

- Static typing
- Run-time efficient

## Python/JS-like

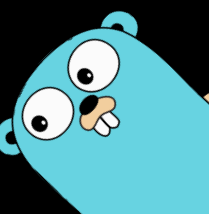
- Readable
- Easy-to-use

## R, R, K like:

- High-performance networking
- High-performant multiprocessing

## Me like:

- Cute logo



# Performance:

Using [the Computer Benchmarks Game](#), the team of researchers tested these languages by compiling/executing such programs using the state-of-the-art compilers, virtual machines, interpreters, and libraries. They then analyzed the performance of the different implementation considering three variables: execution time, memory consumption and energy consumption.

# Stats for nerds:

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64

# Stats for nerds:

Table 5. Pareto optimal sets for different combination of objectives.

Time & Memory	Energy & Time	Energy & Memory	Energy & Time & Memory
C • Pascal • Go	C	C • Pascal	C • Pascal • Go
Rust • C++ • Fortran	Rust	Rust • C++ • Fortran • Go	Rust • C++ • Fortran
Ada	C++	Ada	Ada
Java • Chapel • Lisp • Ocaml	Ada	Java • Chapel • Lisp	Java • Chapel • Lisp • Ocaml
Haskell • C#	Java	OCaml • Swift • Haskell	Swift • Haskell • C#
Swift • PHP	Pascal • Chapel	C# • PHP	Dart • F# • Racket • Hack • PHP
F# • Racket • Hack • Python	Lisp • Ocaml • Go	Dart • F# • Racket • Hack • Python	JavaScript • Ruby • Python
JavaScript • Ruby	Fortran • Haskell • C#	JavaScript • Ruby	TypeScript • Erlang
Dart • TypeScript • Erlang	Swift	TypeScript	Lua • JRuby • Perl
JRuby • Perl	Dart • F#	Erlang • Lua • Perl	
Lua	JavaScript	JRuby	
	Racket		
	TypeScript • Hack		
	PHP		
	Erlang		
	Lua • JRuby		
	Ruby		

# Is it used?

Some notable **open-source** applications written in Go include:

- Caddy, an open source HTTP/2 web server with automatic HTTPS capability.
- CockroachDB, an open source, survivable, strongly consistent, scale-out SQL database.
- **Docker**, a set of tools for deploying Linux containers
- Ethereum, The *go-ethereum* implementation of the Ethereum Virtual Machine blockchain for the *Ether* cryptocurrency
- **InfluxDB**, an open source database specifically designed to handle time series data with high availability and high performance requirements.
- Juju, a service orchestration tool by Canonical, packaging of Ubuntu Linux
- **Kubernetes** container management system
- OpenShift, a cloud computing platform as a service by Red Hat
- Snappy, a package manager for Ubuntu Touch developed by Canonical.
- **Terraform**, an open-source, multiple cloud infrastructure provisioning tool from HashiCorp.

Other notable **companies** and sites using Go include:

- **Dropbox**, who migrated some of their critical components from Python to Go
- **Google**, for many projects, notably including download server dl.google.com
- **MongoDB**, tools for administering MongoDB instances
- **Netflix**, for two portions of their server architecture
- **Twitter**, a variety of micro-services in its Enterprise Cloud OS
- Plug.dj, an interactive online social music streaming website
- SendGrid, Boulder, Colorado-based transactional email delivery and management service
- **SoundCloud**, for "dozens of systems"
- Splice, for the entire backend (API and parsers) of their online music collaboration platform
- ThoughtWorks, some tools and applications for continuous delivery and instant messages (CoyIM)
- **Twitch**, for their IRC-based chat system (migrated from Python)
- **Uber**, for handling high volumes of geofence-based queries

**YES**



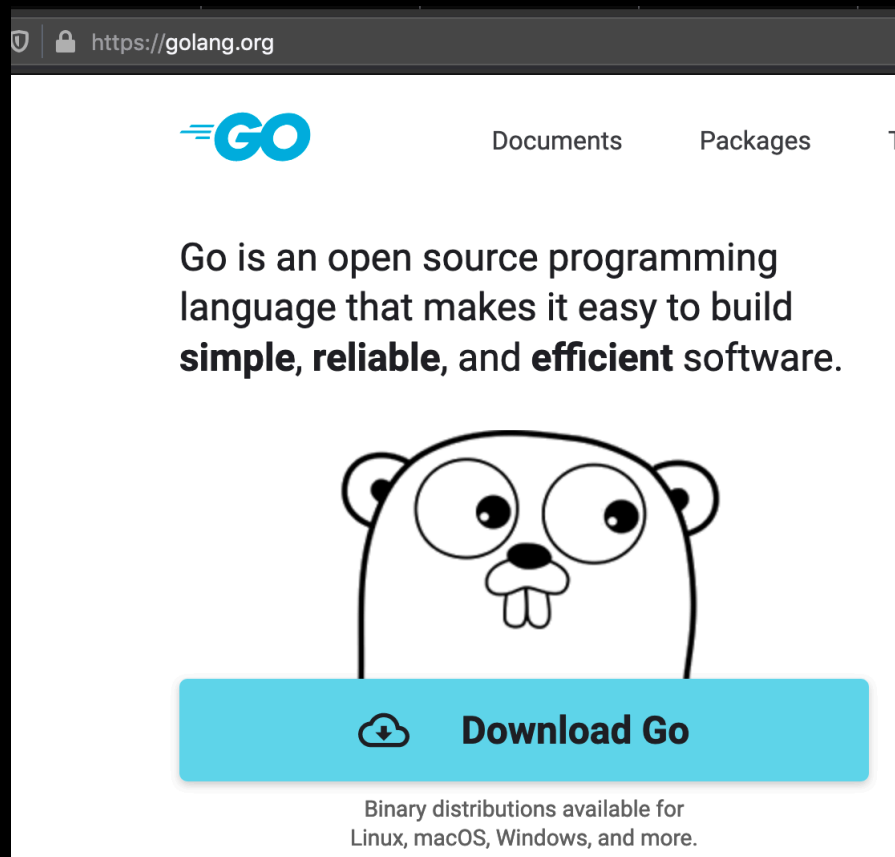
# Package deal

What you get when you download Go

Go (standard library)

Go CLI (build, format, test, manage packages)

Go Docs



# "Main" function

## EDITOR

```
package thisFunctionsPackage
```

```
import "packagename"
```

```
func main() {  
    package.function(parameter1, parameter2)  
}
```

## TERMINAL

```
» go fmt file.go
```

```
» go build file.go
```

```
» ./file
```

OR

```
» go run file.go
```

# Variable declaration

```
var name type  
name = value
```

Type inference

```
name := value
```

```
var number int = 42  
inferredInteger := 42
```

```
aString := "some characters"  
aFloat := 0.42
```

Multiple assignments:

```
name, age := "Nils", 25
```

# Arrays & Slices

## Array

```
numbers := [6]int{3, 16, -2, 10, 23, 12}  
words := [6]string{"one", "two", "3"}
```

—> fixed size, holds 1 specified type

```
for i, number := range numbers {  
    if number >= 10 && number <= 20 {  
        fmt.Println(i, "-", number)  
    }  
}
```

# Loops & Conditional statements

```
if condition == case {  
    pkg.fn(parameter)  
} else {  
    something = "assigned"  
}
```

```
for i := 1; i <= 10; i++ {  
    repeat(stuff)  
}
```

```
var1, var2 := value, value2
```

```
if condition == case {  
    pkg.fn(parameter)  
} else {  
    something = assigned"  
}
```

```
for i := 1; i <= 10; i++ {  
    repeat(stuff)  
}
```