

Golang #5

TO GO
or
NOT TO GO

Vu Nguyen
vu.nguyen@will.vn



Go is an optional language released in 2012

*(It does not force you using it like
Java for Android or Obj-C for iOS)*

Why is it so popular today?

Search

stars:>0

Repositories

22,463

We've found 22,463 repository

<> Code

! Issue

👤 User

3 / 2015

Languages

JavaScript	371,865	
Ruby	245,139	wcong/ants-go
Java	213,398	ants in go
Python	204,142	Updated 5 minutes ago
PHP	167,570	
C	97,485	
C++	88,020	hkwi/gopenflow
Objective-C	62,205	golang openflow implementation
Shell	58,670	Updated 10 minutes ago
C#	57,874	

Search

stars:>0 language:Go

Repositories

33,058

We've found 33,058 repository

<> Code

! Issue

👤 User

12 / 2015

Languages

JavaScript	469,652	
Java	266,288	golang/go
Ruby	259,762	The Go programming language
Python	250,552	Updated 17 minutes ago
PHP	195,229	
C	115,697	
C++	110,723	kubernetes/kubernetes
Objective-C	76,025	Container Cluster Manager from Google
C#	75,916	Updated 20 minutes ago
Shell	73,682	



Who are switching to Go?

(definitely not mobile devs)

	Found	Starting stack	Trends
Twitter	2006	Ruby on Rails	Write new services in Golang https://blog.twitter.com/2015/handling-five-billion-sessions-a-day-in-real-time
Dropbox	2008	Python	Migrate performance-critical backends to Golang https://blogs.dropbox.com/tech/2014/07/open-sourcing-our-go-libraries/
GitLab	2011	Ruby on Rails	Ruby on Rails Partially use Golang https://gitlab.com/gitlab-org/gitlab-git-http-server
Parse	2011	Ruby on Rails	Golang http://blog.parse.com/learn/how-we-moved-our-api-from-ruby-to-go-and-saved-our-sanity/
Koding	2011	NodeJs	Golang https://www.quora.com/Why-did-Koding-switch-from-Node-js-to-Go

Why Go?

1. Single binary deployment
2. Minimal language
3. Easy concurrency
4. Full development environment
5. Multi-arch build
6. Low-level interface
7. Getting started quickly



Nope. Only 1 reason.



Nope. Only 1 reason.

It just works!

Remember the day
when you wrote your first program in University.

```
int main(int argc, const char* argv[]) {  
    printf("%s", "Hello world");  
    return 0;  
}
```

```
$ ./hello  
Hello world
```



Then things get so complex...

Then things get so complex...

What is “AbstractUniversalModelFactoryBuilder” ?

When will we use “abstract class” or “interface” ?

Hey, “callback” or “promise” or “async.js” or “yield”?

How to run your app on multiple-cores computers ? (hint: Node.js “cluster”)

How to correctly install all these dependencies?

“MVC” or “ORM” or “EntityFramework” or “name-your-fancy-framework” ?

Why my database got “undefined” instead of my beautiful numbers?

Why did you use “tab” instead of “4 spaces” ?



Why get into trouble?



Why get into trouble?

Programming languages are tools to
build my beautiful applications.

Nothing more!

Why people create so many things to simplify life of developers?

- Create applications without writing code.
- Build real-time mobile applications without server code.
- Automatically scale up without manually config.
- ORM, frameworks, and IDE.
- ...



Because development is hard.

A vertical bar on the left side of the slide, composed of several colored rectangular segments: red, orange, yellow, green, blue, and purple.

Because development is hard.

Life is short.

Keep building your awesome applications.




Just Go!

Just Go!

```
import "fmt"
func main() {
    fmt.Println("Hello world!")
}
```

```
$ ./hello
Hello world!
```

- 
1. Cross platform build
 2. Garbage collector
 3. Run on multiple-core by default
 4. Easy to learn and write
 5. Consistent coding style, easy to read others' code
 6. Super easy deployment and config
 7. Good and consistent performance
 8. No more crazy “**AbstractUniversalFactory...**”
 9. No more OOP, ORM, fancy frameworks, ...
Just write code that matter.

Writing a web server

```
import (  
    "net/http"  
    "fmt"  
)  
  
func handler(w http.ResponseWriter, r *http.Request) {  
    fmt.Fprintln(w, "Hello World!")  
}  
  
func main() {  
    http.HandleFunc("/", handler)  
    http.ListenAndServe(":8080", nil)  
}
```



Go is a language for engineers

- Go make development life simpler but do not try to hide nasty things.
- You still need your computer science knowledge.

When not Go?

1. Mobile development
2. Web development
3. Game development
4. Data scientist
5. Low-level drivers
6. Performance critical code
7. Prototyping applications
8. MVC applications
9. Shared host (PHP, ASP.NET)

When not Go?

1. Mobile development → Java, Obj-C, .NET
2. Web development → JavaScript
3. Game development → Game engines
4. Data scientist → Python, R
5. Low-level drivers → C, Rust
6. Performance critical code → C, C++, Rust
7. Prototyping applications → Node.js
8. MVC applications → PHP, ASP.NET, Ruby on Rails
9. Shared host (PHP, ASP.NET) → Wordpress!

When Go?

1. Distributed environment
Server development, web services, api
2. Portable
Command line tools
3. You care about team productivity
and good performance & quality.



Golang #5

THANK YOU

Vu Nguyen
vu.nguyen@will.vn