The Go features I can't live without, 2nd round

Golang Brno meetup #2
16 June 2016

Rodolfo Carvalho
Red Hat

Previously

goo.gl/ZkHw4X

1. **Simplicity**
2. Single dispatch
3. Capitalization
4. gofmt
5. godoc
6. No exceptions
7. Table-Driven Tests
8. Interfaces

9

First-class functions

In Go, functions are first-class citizens.

They can be taken as argument, returned as value, assigned to variables, and so on.

Trivial, but not all languages have it... Bash nightmares!

You know what, you can even implement methods on function types!

```
// From Go's net/http/server.go

type HandlerFunc func(ResponseWriter, *Request)

func (f HandlerFunc) ServeHTTP(w ResponseWriter, r *Request) {
    f(w, r)
}
```

Example

```
func main() {
    cmd := exec.Command("bash", "-c", "while true; do date && sleep 1; done")
    cmd.Stdout = os.Stdout

    if err := cmd.Start(); err != nil {
        log.Fatal(err)
    }

    timeout := 5 * time.Second
    time.AfterFunc(timeout, func() {
        cmd.Process.Kill()
    })

    if err := cmd.Wait(); err != nil {
        log.Fatal(err)
    }
}
```

Run

10

Fully qualified imports

Easy to tell where a package comes from.

```
package builder

import (
    "fmt"
    "io/ioutil"
    "os"
    "os/exec"
    "time"

    "github.com/docker/docker/builder/parser"
    kapi "k8s.io/kubernetes/pkg/api"

    "github.com/openshift/origin/pkg/build/api"
    "github.com/openshift/origin/pkg/client"
)
```

Compare with Ruby

```
require 'rubygems'
require 'openshift-origin-node/model/frontend/http/plugins/frontend_http_base'
require 'openshift-origin-node/utils/shell_exec'
require 'openshift-origin-node/utils/node_logger'
require 'openshift-origin-node/utils/environ'
require 'openshift-origin-node/model/application_container'
require 'openshift-origin-common'
require 'fileutils'
require 'openssl'
require 'fcntl'
require 'json'
require 'tmpdir'
require 'net/http'
```

Now we need a way to determine where each of those things come from.

Imports in Go

- Plain strings, give flexibility to language spec.

"The interpretation of the ImportPath is implementation-dependent but it is typically a substring of the full file name of the compiled package and may be relative to a repository of installed packages."

- Path relative to GOPATH.
- *PackageName* != *ImportPath*; by convention, the package name is the base name of its source directory.

Making imports be valid URLs allows tooling (`go get`) to automate downloading dependencies.

Note: fully qualified imports doesn't solve package versioning.

11

Static typing with type inference

- Stutter-fee static typing
- Let the compiler type check, not unit tests
- Easier refactorings

```
cmd := exec.Command("yes")

var cmd = exec.Command("yes")

var cmd *exec.Cmd = exec.Command("yes")

var cmd *exec.Cmd
cmd = exec.Command("yes")
```

12

Speed

- Development
- Compilation
- Execution

Just a superficial reason to justify Go's success.

What I like

Go is a compiled language that feels like scripting*.

Minimal boilerplate:

- No need* for config files, build scripts, header files, XML files, etc.
- The only configuration is GOPATH.

* But not all the time...
E.g., big projects like OpenShift can take 330+s to build with Go 1.6 ☹

Fast feedback cycles

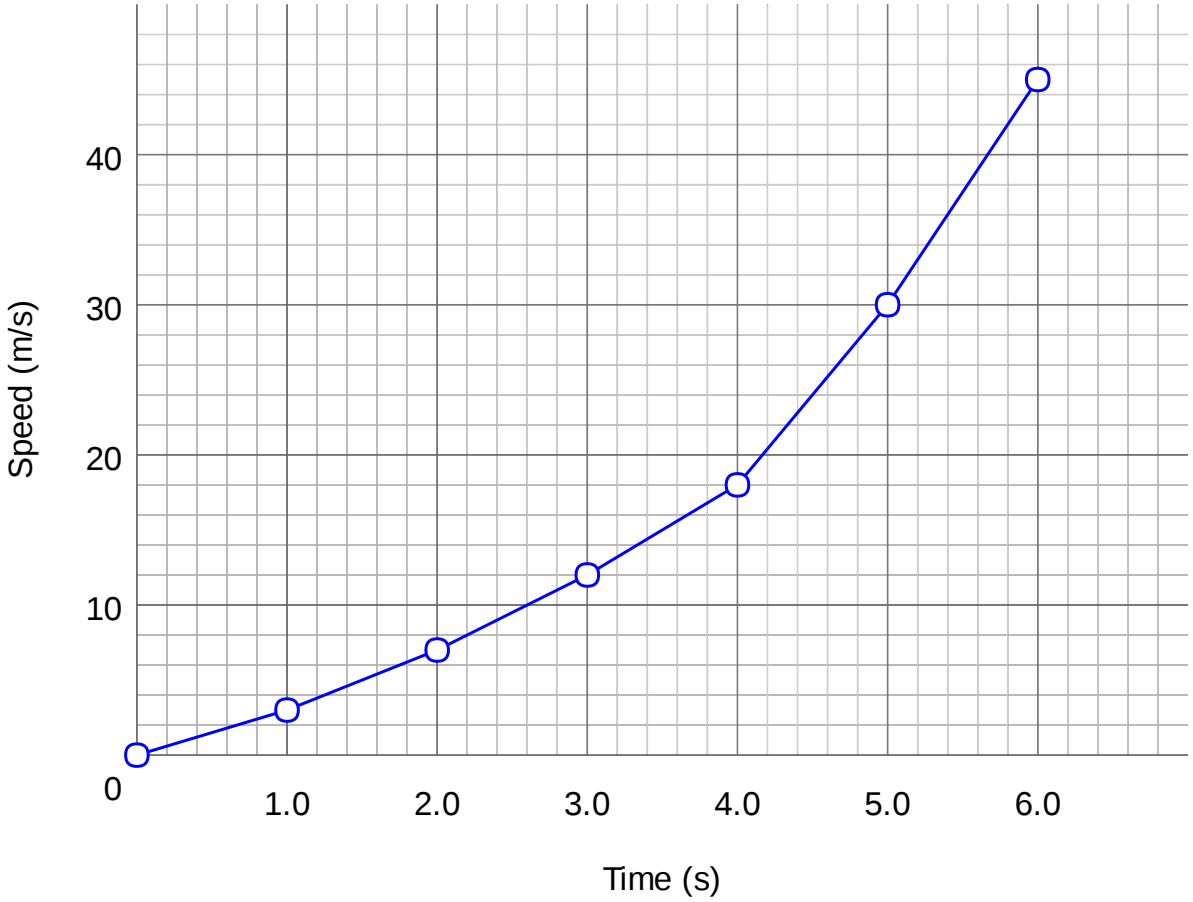
```
package main

import "fmt"

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

Run

How it improved over time



13

Metaprogramming

Wikipedia says: *... the writing of computer programs with the ability to treat programs as their data. It means that a program could be designed to read, generate, analyse or transform other programs, and even modify itself while running.*

- Go has no support for generics, at least yet.
- No macros.

But has:

- reflect package.
- Packages in the stdlib for parsing, type checking and manipulating Go code.
- Go programs that write Go programs and can serve as example: `gofmt`, `goimports`, `stringer`, `go fix`, etc.
- `go generate` tool.

14

Static linking

- The linker creates statically-linked binaries by default.
- Easy to deploy/distribute.
- Big binary sizes.

OpenShift is a single binary with 134 MB today. Includes server, client, numerous command line tools, Web Console, ...

Tale: distributing a Python program.

15

Cross-compilation

Develop/build on your preferred platform, deploy anywhere.

Operating Systems:

- Linux
- OS X
- Windows
- *BSD, Plan 9, Solaris, NaCl, Android

Architectures:

- amd64
- x86
- arm, arm64, ppc64, ppc64le, mips64, mips64le

Easy to use

```
$ GOOS=linux   GOARCH=arm GOARM=7 go build -o func-linux-arm7  func.go
$ GOOS=linux   GOARCH=arm GOARM=6 go build -o func-linux-arm6  func.go
$ GOOS=linux   GOARCH=386         go build -o func-linux-386   func.go
$ GOOS=windows GOARCH=386         go build -o func-windows-386 func.go

$ file func-*
func-linux-386:   ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
                  statically linked, not stripped
func-linux-arm6:  ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV),
                  statically linked, not stripped
func-linux-arm7:  ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV),
                  statically linked, not stripped
func-windows-386: PE32 executable (console) Intel 80386 (stripped to external
                  PDB), for MS Windows

$ du func-*
2184    func-linux-386
2188    func-linux-arm6
2184    func-linux-arm7
2368    func-windows-386
```

16

Go Proverbs

Similar to *The Zen of Python*, there is a lot of accumulated experience and shared knowledge expressed by Go Proverbs.

Not actually a feature, but a "philosophical guidance".

Go Proverbs 1/4

- Don't communicate by sharing memory, share memory by communicating.
- Concurrency is not parallelism.
- Channels orchestrate; mutexes serialize.
- The bigger the interface, the weaker the abstraction.
- Make the zero value useful.

Go Proverbs 2/4

- interface{} says nothing.
- Gofmt's style is no one's favorite, yet gofmt is everyone's favorite.
- A little copying is better than a little dependency.
- Syscall must always be guarded with build tags.
- Cgo must always be guarded with build tags.

Go Proverbs 3/4

- Cgo is not Go.
- With the unsafe package there are no guarantees.
- Clear is better than clever.
- Reflection is never clear.
- Errors are values.

Go Proverbs 4/4

- Don't just check errors, handle them gracefully.
- Design the architecture, name the components, document the details.
- Documentation is for users.
- Don't panic.

Recap

9. First-class functions
10. Fully qualified imports
11. Static typing with type inference
12. Speed
13. Metaprogramming
14. Static linking
15. Cross-compilation
16. Go Proverbs

Thank you

Rodolfo Carvalho
Red Hat
rhcarvalho@gmail.com
http://rodolfocarvalho.net