The State of Go
Where we are on February 2018

Francesc Campoy
VP of Developer Relations at source{d}
Time flies

Go 1.8 is one year old (Happy Birthday!)

Go 1.9 is already 6 months old!

Go 1.10rc1 was released on January 25th.

Go 1.10 is about ✋ to be released!
Notes

The slides are already available on campoy.cat/l/sog110

Most of the code examples won't run except locally and using Go 1.10.

The playground still runs Go 1.9.

👉 do not send issues about the slides not running correctly online!
Agenda

Changes since Go 1.9:

- The Language
- The Ports
- The Tooling
- The Standard Library
- The Performance
- The Community
Changes To The Language
Changes To The Language
Ports
New Ports

source
Notes On Existing Ports

- FreeBSD: requires FreeBSD 10.3 or later
- NetBSD: works but requires NetBSD 8 ... which is not released yet
- OpenBSD: next version will require OpenBSD 6.2
- OS X: next version will require OS X 10.10 Yosemite
- Windows: next version will require Windows 7 (no more XP or Vista)
- 32-bits MIPS have now a new GOMIPS variable (**hardfloat** | **softfloat**)
One More Note On Existing Ports

It's rare that I laugh out loud while reading GitHub issues.

And even after Go 1.12 comes out, you can keep using Go 1.10, we just won't fix bugs in it. But if you're happy with it (or Go 1.9 or whatever version), great. You won't get security fixes, but if you are running XP you're not worried about that.

😊 3
Changes To The Tooling
Changes To The Tooling

In two words: easier and faster.
Easier set-up

GOPATH became optional in Go 1.8.

GOROOT is now optional too, deduced from the binary path.

A new variable GOTMPDIR was added to control where temporary files are created.
Faster tools via caching

- go install now caches the result of compiled packages.
- go install and go build are **much** faster in general as a result
- you won't need go build -i anymore!

It seems the pkg directory might eventually disappear!
Testing

Also caches results, everything is faster

```bash
➜  go test strings
ok strings (cached)
```

In order to bypass the cache use `-count=1`

```bash
➜  go test -count=1 strings
ok strings 0.295s
```

Also runs `vet`, some of your tests might fail.

Also:

- `coverprofile` can be done over many tests too
- `new -failfast` and `-json` flags
A Small Detour
Three-Index Slicing

Did you know you can use three values for slicing?

```go
text := []byte("Hello FOSDEM!")
fmt.Printf("text:  %s", desc(text))

hello := text[0:5]
fmt.Printf("hello: %s", desc(hello))

hello = append(hello, '#')
fmt.Printf("hello: %s", desc(hello))

fmt.Printf("text:  %s", desc(text))
```

Run
Three-Index Slicing (cont.)

You can control the capacity of the resulting slice.

```go
text := []byte("Hello FOSDEM!")
fmt.Printf("text: %s", desc(text))

hello := text[0:5:5]
fmt.Printf("hello: %s", desc(hello))

hello = append(hello, '#')
fmt.Printf("hello: %s", desc(hello))

fmt.Printf("text: %s", desc(text))
```

Run
Small change in formatting of three-index slicing expressions.

Before:

\[ a[i : j : k] \]

Now:

\[ a[i : j : k] \]

This might break some of your CI tests (it broke some of mine).
Changes To The Standard Library
Changes To The Standard Library

No new packages with Go 1.10

Trivia: Do you remember which new package was added with Go 1.9?
Changes to bytes

Fields, FieldsFunc, Split, and SplitAfter limit the capacity of the returned slices.

```go
text := []byte("Hello FOSDEM!")
fmt.Printf("text:  %s", desc(text))

hello := bytes.Fields(text)[0]
fmt.Printf("hello: %s", desc(hello))

hello = append(hello, '#')
fmt.Printf("hello: %s", desc(hello))

fmt.Printf("text:  %s", desc(text))
```

playground
Changes to flags

This is minor, but I am very happy about it!

stuff := flag.Int("s", 0, "some other stuff
it's long to explain")
z := flag.Int("z", 42, "some number")
flag.Parse()

Before 😞

-s int
    some other stuff
it's long to explain
-z int
    some number (default 42)

Now 😎

-s int
    some other stuff
    it's long to explain
-z int
    some number (default 42)
Changes to go/doc

For a type T, functions returning slices of T, *T, or **T are now linked to T.

Those functions now appear in the Funcs list of the type, not the package.

Example:

```go
package things

// Thing is stuff.
type Thing struct{}

// NewThing returns a new thing.
func NewThing() *Thing { return nil }

// ManyThings returns many new things.
func ManyThings() []Thing { return nil }
```
Changes to go/doc (cont.)

Before 😞

```go
package things // import "github.com/campoy/talks/go1.10/things"

func ManyThings() []Thing
type Thing struct{
    func NewThing() *Thing

Now 😎

```go
package things // import "github.com/campoy/talks/go1.10/things"

type Thing struct{
    func ManyThings() []Thing
    func NewThing() *Thing
```
Changes to text/template

New `{{break}}` and `{{continue}}` for `{{range}}`.

```go
var tmpl = template.Must(template.New("example").Funcs(template.FuncMap{
    "even": func(x int) bool { return x%2 == 0 },
}).Parse(`
{{ range . }}
{{ . }}
{{ if even . -}}
    even
    {{ continue }}
{{ end -}}
odd
{{ if eq . 5 }}
    {{ break }}
{{ end }}
{{ end }}
```

*Note:* Interestingly, this is not implemented in the `html` package.
strings

I'm sure you've written this kind of code before.

```go
var buf bytes.Buffer
fmt.Fprintln(&buf, "Hello, FOSDEM gophers!")
fmt.Printf(buf.String())
```

But there's some issues with it.

String creates allocations since it converts `[]byte` to `string`.

There could be a better and simpler way to do this.

```go
var b strings.Builder
fmt.Fprintln(&b, "Hello, FOSDEM gophers!")
fmt.Printf(b.String())
```

This uses unsafe to avoid copies in the creation of strings.
strings.Builder

When you're creating many strings, it is definitely worth it.

```go
for i := 0; i < 10000; i++ {
    fmt.Fprintf(w, "😊")
    out = w.String()
}
```

Benchmark results:

```
$ go test -bench=. -benchmem
goos: darwin
goarch: amd64
pkg: github.com/campoy/talks/go1.10/strings
BenchmarkBuffer-4            100          20861915 ns/op        215641272 B/op     10317 allocs/op
BenchmarkBuilder-4          3000            535081 ns/op          153647 B/op         22 allocs/op
PASS
ok      github.com/campoy/talks/go1.10/strings  3.626s
```
strings.Builder 🙄

When you're creating many strings, it is definitely worth it.

```go
for i := 0; i < 10000; i++ {
    fmt.Fprintf(w, "😊")
    // out = w.String()
}
```

Benchmark results:

```
$ go test -bench=. -benchmem
goos: darwin
goarch: amd64
pkg: github.com/campoy/talks/go1.10/strings
BenchmarkBuffer-4    3000    525691 ns/op    152056 B/op    11 allocs/op
BenchmarkBuilder-4   3000    626132 ns/op    153647 B/op    22 allocs/op
PASS
ok      github.com/campoy/talks/go1.10/strings  4.072s
```
oh my gopher!
sure ... why not
unicode

roar
mind blown
and the unicode character we all wanted

the character we deserve
Performance Changes
Runtime Performance

After running all the benchmarks on the standard library on go1.9.3 vs go1.10rc1:

- nothing changed

```
$ benchstat go1.9.txt go1.10.txt  | grep -v "\-"
```
Compiler Performance

Compiling the standard library is **10% faster**!

```
$ benchstat go1.9.3.txt go.1.10rc1.txt
name          old time/op   new time/op   delta                     
Template       234ms ± 4%  231ms ± 4%  ~ (p=0.101 n=10+8)   
Unicode        107ms ± 1%  109ms ± 6%  ~ (p=0.211 n=9+10)   
GoTypes        742ms ± 2%  744ms ± 2%  ~ (p=0.905 n=9+10)   
Compiler       3.50s ± 3%  3.54s ± 5%  ~ (p=0.393 n=10+10)  
SSA            6.95s ± 4%  9.04s ± 5%  +29.98% (p=0.000 n=10+10) 
Flate          149ms ± 2%  147ms ± 5%  -1.53% (p=0.035 n=10+9)  
GoParser       189ms ± 3%  183ms ± 3%  -3.44% (p=0.002 n=9+9)  
Reflect        476ms ± 5%  489ms ± 6%  +2.90% (p=0.043 n=10+10) 
Tar            134ms ± 1%  220ms ± 3%  +64.14% (p=0.000 n=9+10)  
XML            258ms ± 6%  266ms ± 6%  +2.90% (p=0.043 n=10+10) 
StdCmd          19.1s ± 1%  17.1s ± 3%  -10.57% (p=0.000 n=10+10) 
```

Following [https://golang.org/x/tools/cmd/compilebench](https://golang.org/x/tools/cmd/compilebench).

Run on a Google Compute Engine instance with 8 cores.
Garbage Collector History in Tweets
Amazing GC pause time improvements in Go 1.5.
Brian Hatfield @brianhatfield · 28 Jan 2016
They did it again in Go 1.6 RC 1!
Excited to canary Go 1.7! Continued improvement in GC pause, and improvements in various request latencies/perf!
(Trying out Go 1.8 beta 1!)

Go 1.7.3

App Warmup

Go 1.8 (beta 1)
Brian Hatfield @brianhatfield · 23 Aug 2017
1.9rc2 canary: sub-millisecond pause time GC (18GB heap). Same as 1.8.3.

If you're not on 1.8.3, upgrade or try 1.9rc2.
and finally, go 1.10
and finally, go 1.10

Francesc
@francesc

Replying to @brianhatfield

waiting for the 1.10rc1 stats for my "State of Go" talk ... 😊

12:49 PM - 2 Feb 2018

3 Likes
and finally, go 1.10

**Brian Hatfield** 11:05 PM
Figured I'd find you here 😊

**Francesc Campoy** 11:05 PM
oh hey!

**Brian Hatfield** 11:07 PM
Hey!
Running a canary now but it's gonna take some extra time for reasons 😞
Initial impressions seem to not be different however.

**Francesc Campoy** 11:11 PM
oh, that’s what I expected indeed 😊

**Francesc Campoy** 11:37 PM
do you mind if I use a screenshot of this conversation for my talk?
feel free to use your next message to say hi to the audience 😊

**Brian Hatfield** 11:39 PM
hahaha! hi, FOSDEM 2018!! Sure, screenshot away 😊
and then this morning ...
and the this morning ...

Go 1.10rc1 canary: no significant performance change observations - GC pause, request latency, CPU usage all effectively the same as 1.9.

4:39 PM - 2 Feb 2018
A couple more changes too

Go 1.10 release notes (DRAFT)
Women Who Go

26 chapters already - 10 more than last year! www.womenwhogo.org
Women Who Go Leaders

CEO / Global Visionary & Lead
Maartje Eyskens

CFO / Financial Director
Verónica López

Head of New Chapters
Daniela Petruzalek

Head of Support
Carolyn Van Slyck
Go meetups

Gophers all around the world! (367 meetups on go-meetups.appspot.com)
Conferences:

- Go Devroom FOSDEM Today and here! 🎉
- GopherCon India - March in Pune, India
- GopherCon Russia - March in Moscow, Russia
- GoSF - March in San Francisco, USA
- GothamGo - April in New York, USA
- GopherCon SG - May in Singapore
- GopherCon Europe - June in Reykjavik, Iceland
- GopherCon Denver - August in Denver, USA
- GopherCon Brasil - September in Florianópolis, Brazil
- GoLab - October in Florence, Italy
- dotGo - March 2019 in Paris, France
<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Speaker(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:30-11:00</td>
<td>The State of Go</td>
<td>Francesc Campoy</td>
</tr>
<tr>
<td>11:00-11:30</td>
<td>Advanced Go debugging with Delve</td>
<td>Derek Parker</td>
</tr>
<tr>
<td>11:30-12:00</td>
<td>Testing and Automation in the Era of Containers (with Go)</td>
<td>Verónica López</td>
</tr>
<tr>
<td>12:00-12:30</td>
<td>Networking deepdive</td>
<td>Michael Hausenblas</td>
</tr>
<tr>
<td>12:30-13:00</td>
<td>Uppspin and a future of the Internet</td>
<td>Gildas Chabot</td>
</tr>
<tr>
<td>13:00-13:30</td>
<td>Dep Deep Dive!</td>
<td>Sam Boyer</td>
</tr>
<tr>
<td>13:30-14:00</td>
<td>Networking Swiss Army Knife for Go</td>
<td>Roman Mohr</td>
</tr>
<tr>
<td>14:00-14:30</td>
<td>The case for interface()</td>
<td>Sam Whited</td>
</tr>
<tr>
<td>14:30-15:00</td>
<td>Google's approach to distributed systems observability for Go</td>
<td>Jaana Dogan (JBD)</td>
</tr>
<tr>
<td>15:00-15:30</td>
<td>Creating GopherJS Apps with gRPC-Web</td>
<td>Johan Brandhorst</td>
</tr>
<tr>
<td>15:30-16:00</td>
<td>Building and testing a distributed data store in Go</td>
<td>Matt Bostock</td>
</tr>
<tr>
<td>16:00-16:30</td>
<td>Computer Vision Using Go And OpenCV</td>
<td>Ron Evans</td>
</tr>
<tr>
<td>16:30-17:00</td>
<td>Make your Go go faster!</td>
<td>Bryan Boreham</td>
</tr>
<tr>
<td>17:00-17:30</td>
<td>Optimising performance through reducing memory allocations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distributing DevOps tools using GoLang and Containers, for Fun and Profit!</td>
<td>Lucy Davinhart</td>
</tr>
<tr>
<td>17:30-18:00</td>
<td>AMENDMENT DNA sequencing performance in Go, C++, and Java</td>
<td>Pascal Costanza</td>
</tr>
<tr>
<td>18:00-19:00</td>
<td>Go Lightning Talks</td>
<td>Francesc Campoy, Maartje Eyskens</td>
</tr>
</tbody>
</table>

*Come speak!*
Enjoy the rest of the day!

Gopher by the amazing Ashley McNamara
Thank you

Francesc Campoy
VP of Developer Relations at source{d}
@francesc
campoy@golang.org
https://sourced.tech