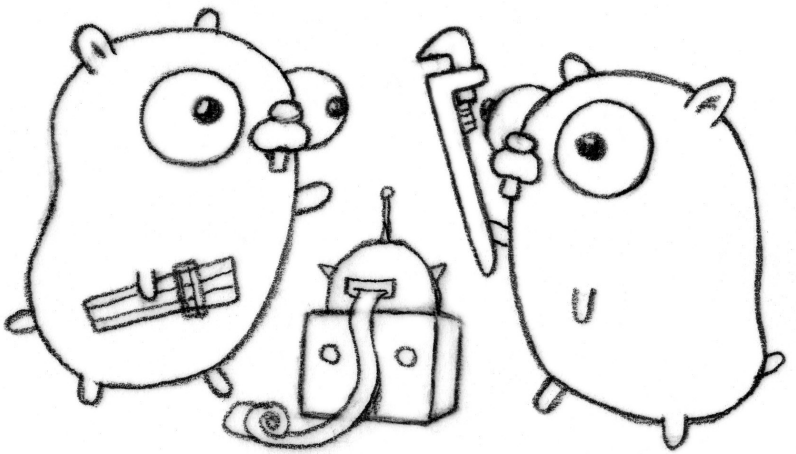
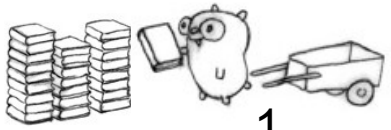
Go language banter

Good design

Wow

Much wonderful





Entry points

nice online tutorial
74 steps

tour.

online compiler

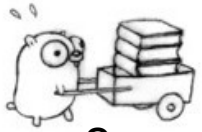
play.

golang.org

all documentations

doc.

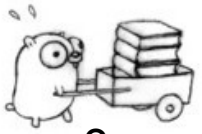
„Effective Go“ : a must-read

Language features

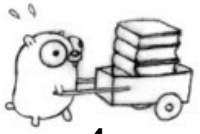
- Compiled
- Statically-typed
- Fast compilation, fast execution
- Garbage collection
- **Interfaces**
- Pointers
- **Slices**
- Maps

- **Concise syntax**
- readable
- Easy to learn
- **Concurrency** with goroutines and **Channels**
- Functions are first-class
- Reflection


Community

- Designed by a **small committee** of rockstars
- Lots of official packages
  - Networking
  - Encoding
  - Templating
  - etc.
  - etc.
- Lots of user packages
- Already used in production
- **Available on Google App Engine**
- Mailing list
- Forums

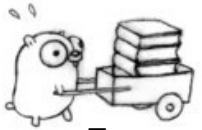

Multiple results

A function can return any number of values

http://tour.golang.org/#9

http://play.golang.org/p/0po_GGkrLz

```
func donaldsNephews() (string, string, string) {
   return "Huey", "Dewey", "Louie"
}

func main() {
   a, b, c := donaldsNephews()
...
```

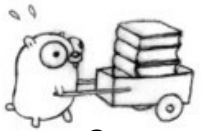




Variable declaration and initializer

- Declaration **var**
- Assignment **=**
- Declaration, type inference and initialization **:=**

```
var i, j int = 1, 2

i = 4

k := 3

c, python, java :=
true, false, "no!"
```

http://tour.golang.org/#13

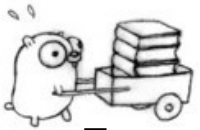




For

```
for i := 0; i < 10; i++ {
    sum += i
}

for sum < 1000 {
    sum += sum
}

for k, v := range myMap {
    fmt.Printf("myMap[%v]=%v \n", k, v)
}
```

- Just **for** !

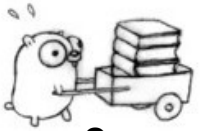
http://tour.golang.org/#18


If

```
if a == b {
    return true
}

if v := math.Pow(x, n); v < lim {
    return v
}
```

http://tour.golang.org/#23

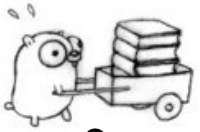




Struct

```
type Vertex struct {
    X int
    Y int
}
```

```
v := Vertex{1, 2}
w := Vertex{
    Y:4,
    X:3,
}
```

http://tour.golang.org/#26

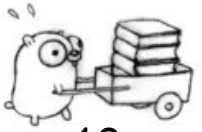




Pointer

```
var q *Vertex

p := Vertex{1, 2}

q = &p

q.X = 1000
```

- Symbols * and **&**
- Indirection through the pointer is transparent.

http://tour.golang.org/#28

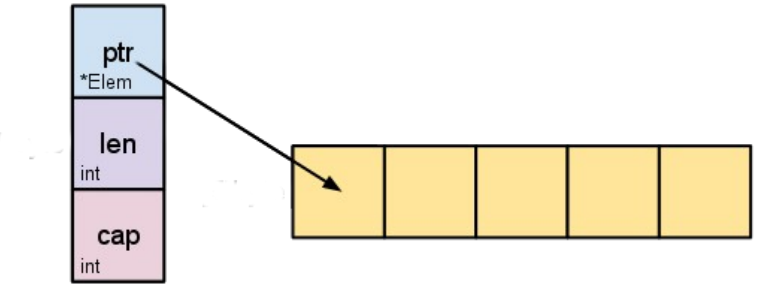




Arrays

- [10]int is a type
- Cannot be resized
- Boring

http://tour.golang.org/#31

Slices



- lightweight
  (like a "smart pointer")
- safe access

http://blog.golang.org/go-slices-usage-and-internals

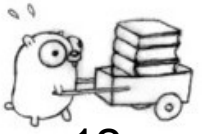


Slicing

```
a := make([]int, 5)

b := make([]int, 0, 5)

p := []int{2, 3, 5, 7}

q := p[1:3]

p = append(p, 11)
```

- len, cap
- shared underlying array
- reallocation auto
  (sometimes needed, sometimes not)

http://tour.golang.org/#32 and following

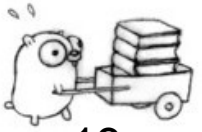




Map

- Built-in type
- Typed keys and values
- Values : any type
- Keys : any comparable type

```
var m map[string]Vertex

m = make(map[string]Vertex)

m["Bell Labs"] = Vertex{
    40.68433, -74.39967,
}

v, ok := m["Bull"]
```

http://tour.golang.org/#39 and following


Function

```
func square(x float64) float64{
  return x*x;
}

hypot := func(x, y float64)
float64 {
    return math.Sqrt(x*x + y*y)
}

z := hypot(3, 4)
```

- First-class
- Typed
- As argument
- As return value
- Named parameters and return values (optional)
- Closures

http://tour.golang.org/#44 and following

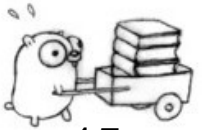


Switch

- Multi-purpose :
  - over variable value
  - over arbitrary conditions (like if-then-else chains)
  - over types

```
os := runtime.GOOS
switch os {
    case "darwin":
        fmt.Println("OS X.")
    case "linux":
        fmt.Println("Linux.")
    default:
        fmt.Printf("%s.", os)
}
```

http://tour.golang.org/#47


Method

http://tour.golang.org/#52

- Receiver { struct / or (preferably) pointer / or slice, map, func ! (as renamed types)

```
type Vertex struct {
    X, Y float64
}

func (v *Vertex) Abs() float64 {
    return math.Sqrt(v.X*v.X + v.Y*v.Y)
}
```

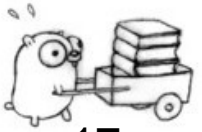


Interface

- Lean OO-style
- Implicit implementation !
- Great decoupling

http://tour.golang.org/#55

```
type Writer interface {
        Write(p []byte) (n int, err error)
}
```

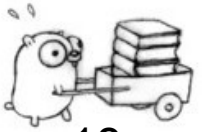

Error

- Idiomatic : return an *error* as last value
  - then the caller checks if the error is nil

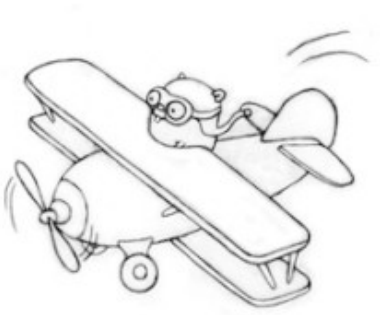
```
x, err := Sqrt(-1)
if err != nil {
    fmt.Println(err)
}
```

http://tour.golang.org/#57

- Or use panic/recover
  - like a try/catch
  - rare
  - **defer** can be useful

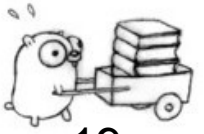




Concurrency



- Concurrency is not parallelism
- A model for software construction :
  Communicating Sequential Processes

http://talks.golang.org/2012/concurrency.slide

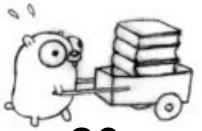




Goroutine

- Concurrent
- Parallel (or not!)
- Lightweight
- Cheap (kind of)

http://tour.golang.org/#65

```
var delay = 100 * time.Millisecond

func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(delay)
        fmt.Println(s)
    }
}

func main() {
    go say("world")
    say("hello")
}
```



Channel

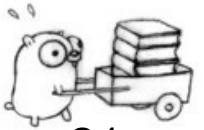
```
func worker(in chan *A, out chan *B)
{
  for {
    a := <- in
    b := transform(a)
    out <- b
  }
}
```

```
in := make(chan *A)
out := make(chan *B)

for i := 0; i < NumWorkers; i++ {
    go worker(in, out)
}
go sendLotsOfWork(in)
receiveLotsOfResults(out)
fmt.Println("Done.")
```

- Producer-consumer
  - Sending values
  - Receiving values
- Proper synchronization
- Blocking, or buffered

http://tour.golang.org/#66

http://play.golang.org/p/a18jzvsFC2

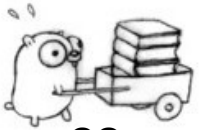


Select

- Wait for event
- Syntax
  - *like a switch*
- Channels
  - input and/or output
- Timeout

```
func fibonacci(c, quit chan int) {
    x, y := 0, 1
    for {
        select {
        case c <- x:
            x, y = y, x+y
        case <-quit:
            fmt.Println("quit")
            return
        }
    }
}
```

http://tour.golang.org/#69



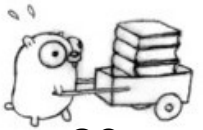
Templating

- Template "compilation"
- Actions
- **Pipelines**

```
{{define "sub"}}
    {{range .}}>> {{.}} {{end}}
{{end}}

*The great GopherBook*
(logged in as {{.User}})

[Most popular]{{template "sub" .Popular}}

[Most active]{{template "sub" .Active}}

[Most recent]{{template "sub" .Recent}}
```

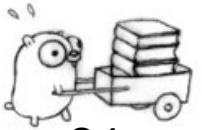
http://golang.org/pkg/text/template

http://golang.org/pkg/html/template



What is missing

- generics
  - *but **interfaces** do the job !*
- i18n
- browser runtime
  - *go is **not** a competitor of javascript for client-side, but translators exist*
- a good debugger
- enforced immutability



Thank you

- Go and have fun
- Go and build servers