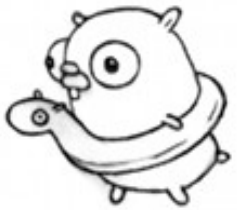
Why you should give it a try

ronny.brendel@tu-dresden.de

http://golang.org

Table of content

- A short history of Go
- What's wrong with current languages?
- How is Go different?
  - The ideas and concepts
  - The syntax/semantics
  - The tools
- Conclusion
- Where to Go from here



History

- Robert Griesemer, Rob Pike and Ken Thompson began working mid 2007
- Full-time project since mid 2008
- Public and open source since November 2009
- In active use at Google since mid 2010
- Go 1 released in March 2012

Current languages

- Designed for the last century – but much has changed since then (multi-core, networks, clusters, huge nr. of libraries)
- Compiling takes (unnecessarily) long
- Dependency jungle (#include, library versions)
- Difficult to understand and get right (e.g. c++'s const and templates, OO is hard)
- Verbosity, legacy problems, ...

Goals

- “The efficiency of a statically-typed compiled language with the ease of programming of a dynamic language.”

- System- / general purpose language
- Statically typed
- Garbage collection
- Good support for concurrency
- Quick compilation

Design principles

- Ease of use:
  - Clean, concise syntax
  - Orthogonal features
  - Convention over configuration
  - No type hierarchies
- Pragmatic
- CSP-inspired concurrency

Hello World < Syntax

- file “helloworld.go”:

```
package main
import (
    "fmt"
    "time"
)
func main() {
    fmt.Println("Hɛllø World!", time.Now())
}
```

- $ go run helloworld.go

```
Hɛllø World! 2013-01-08 16:30:02.732295 +0100 CET
```

Basics < Syntax

- `var x, π *float32`
- `var π = 1.0`
- `π := 1.0`

- `const n = 1024`
- `const (`

```
    i, j int = iota, iota + 1
    k, l // repeats the line above
)
```

- `defer someFunc()`

Basics < Syntax

- `func MyFunc(a, b int) int {}`
- `func MyFunc() (c, d int) {}`

- `func someFunc(a func() int) { return a() }`
- `someFunc(func() int {return 0})`

- `if _, i := MyFunc(); i > 0 {}`
- `if i > 0 {} else if true {} else {}`

- `for i := 0; i < n; i += 1 {}`
- `for {}`

Basics < Syntax

- **switch** x += 1; x {
  case 1:
    // do things
    fallthrough // break is default!
  case 2,3,4:
    // do other things
  default:
    // else
  }
- **switch** {
  case i < 10 && someFunc() == **false**:
    // do things
  }

Values and types < Syntax

- `weekend := []string{"Saturday", "Sunday"}`
- ```
  weekend := map[string]int {
      "Saturday" : 0, "Sunday" : 1
  }
  ```

- ```
  type T struct {
      a, b string
      c bool
  }
  ```
- `x := T{weekend[0], "asdf", false}`

- `type U func(int, int) int`

Methods < Syntax

- ```
  type Point struct {
      X, Y float64 // Upper case means exported
  }
  ```
- ```
  func (p *Point) Scale(s float64) {
      p.X *= s; p.Y *= s
  }
  ```
- ```
  func (p Point) Abs() float64 {
      return math.Sqrt(p.X*p.X + p.Y*p.Y)
  }
  ```
- ```
  x := &Point{ 3, 4 }
  ```
- ```
  x.Scale(5) // . works on pointers and normal types
             // the same way
  ```
  ```

Interfaces < Syntax

- `type Magnitude interface {`
  `  Abs() float64`
  `}`
- `var m Magnitude`
- `m = Point{1, 2}`
- `mag := m.Abs()`
- `type Point3 struct { X, Y, Z float64 }`
- `func (p Point3) Abs() float64 {`
  `  return math.Sqrt(p.X*p.X + p.Y*p.Y + p.Z*p.Z)`
  `}`
- `m = Point3{ 3, 4, 5 }`
- `mag += m.Abs()`

Concurrency < Syntax

- `var chan string`
- `c = make(chan string)`
- `c <- "hello"`

- `// in a different goroutine`
- `greeting := <-c`

- ```
  func Pass(a chan<- int, b <-chan int) {
      a <- b
  }
  ```
  ```

Concurrency < Syntax

- `x := longCalculation(17) // blocks for a long time`

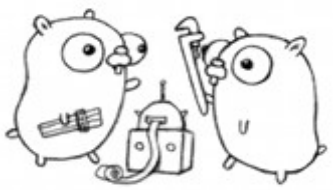
- `// instead try`
- `c := make(chan int)`

- `go func (a int, c chan<- int) {`
  `c <- longCalculation(a)`
- `}(17, c)`

- `// do something while longCalculation runs`
- `x := <-c`

Much more... < Syntax

- Slices
- Pointers (but no pointer arithmetic)
- Variadic Arguments
- Packages
- Reflection
- Iterating
- Testing
- Directory- / project structure



Tooling

- “go” command
  - go doc
  - go fmt
  - go build
  - go get
  - go install
- gdb debugging support
- Syntax-highlighting for many editors

Conclusion

- Promising attempt to create to a better/modern general-purpose language
- Adoption is growing
- Wealth of packages exists (e.g. visit http://godashboard.appspot.com/)

- No major changes since Go 1
- Wait and observe how people use it ... someday maybe Go 2

Where to Go from here

- http://tour.golang.org/
- http://golang.org/doc/
- http://talks.golang.org/

- #go-nuts on irc.freenode.net

Sources

- [1] Golang Docs, Website, 13-01-07
  http://golang.org/doc/
- [2] Go: a simple programming environment, Presentation, 13-01-07
  http://vimeo.com/53221558
- [3] Wikipedia: Go (Programming Language), Website, 13-01-07
  http://en.wikipedia.org/wiki/Golang
- [4] The Go Programming Language, Slide Deck, 13-01-07
  http://talks.golang.org/2009/go_talk-20091030.pdf
- [5] Wikipedia: CSP, Website, 13-01-08
  http://en.wikipedia.org/wiki/Communicating_sequential_processes

Thank you!

Slides are available at
http://automaton2000.com/go-slides.pdf