Write your own Go compiler

A talk by Elliott Stoneham at FOSDEM on 2nd February 2014

Why?

Go is addictive, I want it on the client-side too…

overview

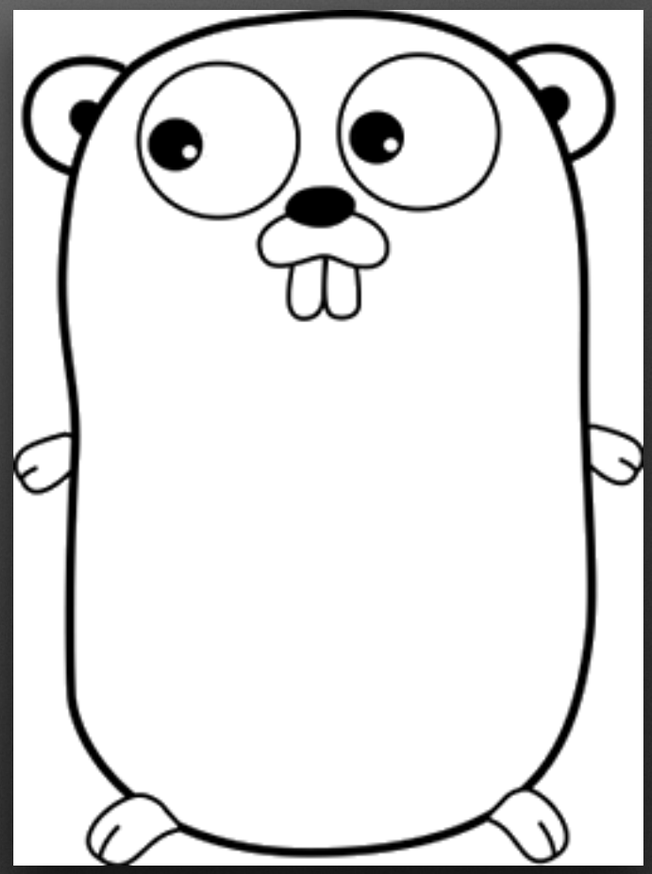
- The Go language and compiler projects
- The go.tools repository and “mutant” Go compilers
- Prospects for “mutant” Go client-side systems
- Signposting the design decisions involved

Success for me today?

You try out, help improve, or make your own “mutant” Go compiler.

Go language specification

- The smallest size possible for the functionality
- Easy and fun to learn, read, write, maintain …
- … and implement for a new target environment
- => a very portable language



active Go compiler projects

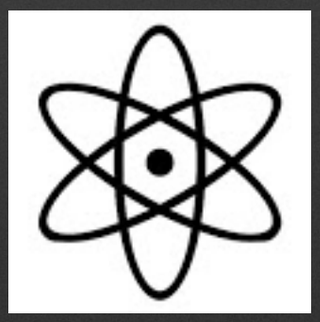
GOLANG	CLOSED SOURCE	OPEN SOURCE
go	Tulgo	llgo
gccgo		GopherJS
		TARDIS Go

Go 1.3+ Compiler Overhaul

- The Go compiler today is written in C. It is time to move to Go.
- We plan to translate the existing compilers from C to Go by writing and then applying an automatic translator. The conversion will proceed in phases, starting in Go 1.3 but continuing into future releases.
- From a document by Russ Cox, December 2013: http://golang.org/s/go13compiler

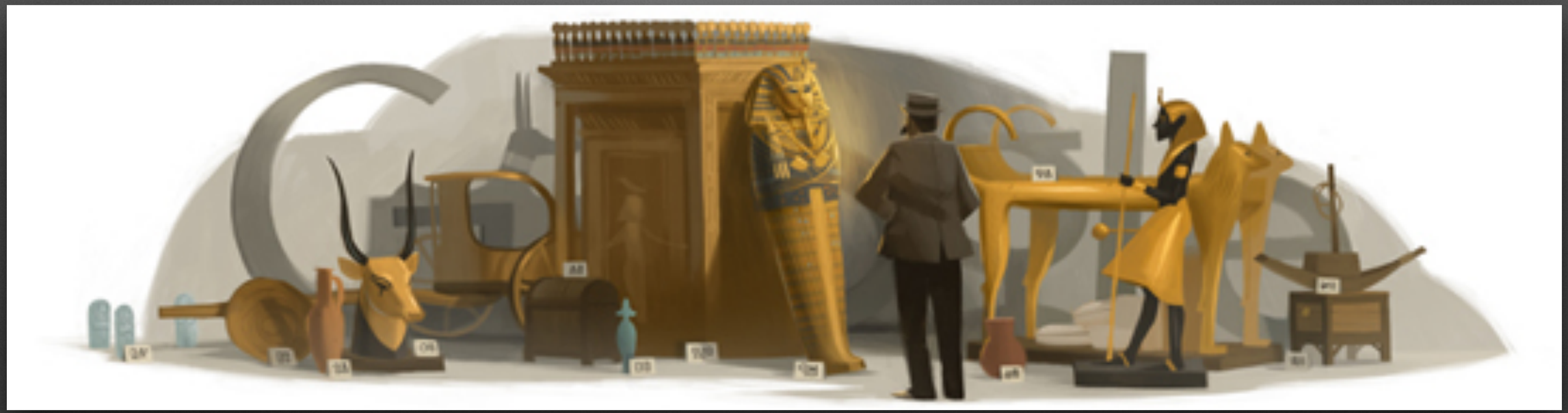
Tulgo blog
http://atom-symbol.rhcloud.com/a/

- [2014-01-23] Tulel introduction
- [2014-01-17] Tulgo - New register allocation pass
- [2013-12-29] Tulgo - First steps in partial rewrite of the core algorithm
- [2013-12-19] Tulgo - Caching the results of DIV instruction in function fmt.(*fmt).integer()
- [2013-12-17] Tulgo - Removing safety costs from function fmt.(*cache).get()
- [2013-12-14] Tulgo - fmt.Fprintf() in a loop
- [2013-11-27] Tulgo - Some intermediate results



“Mutant” open-source Go compilers

	AUTHORS	PROJECT FIRST PUBLISHED	GITHUB STARS	TARGET RUNTIMES
LLGO	Andrew Wilkins + 9 others	October 2011	400+	LLVM
GOPHERJS	Richard Musiol	November 2013	200+	JavaScript
TARDISGO	Elliott Stoneham	January 2014	<10	JS, Flash, Java, C++, C#, PHP & Neko

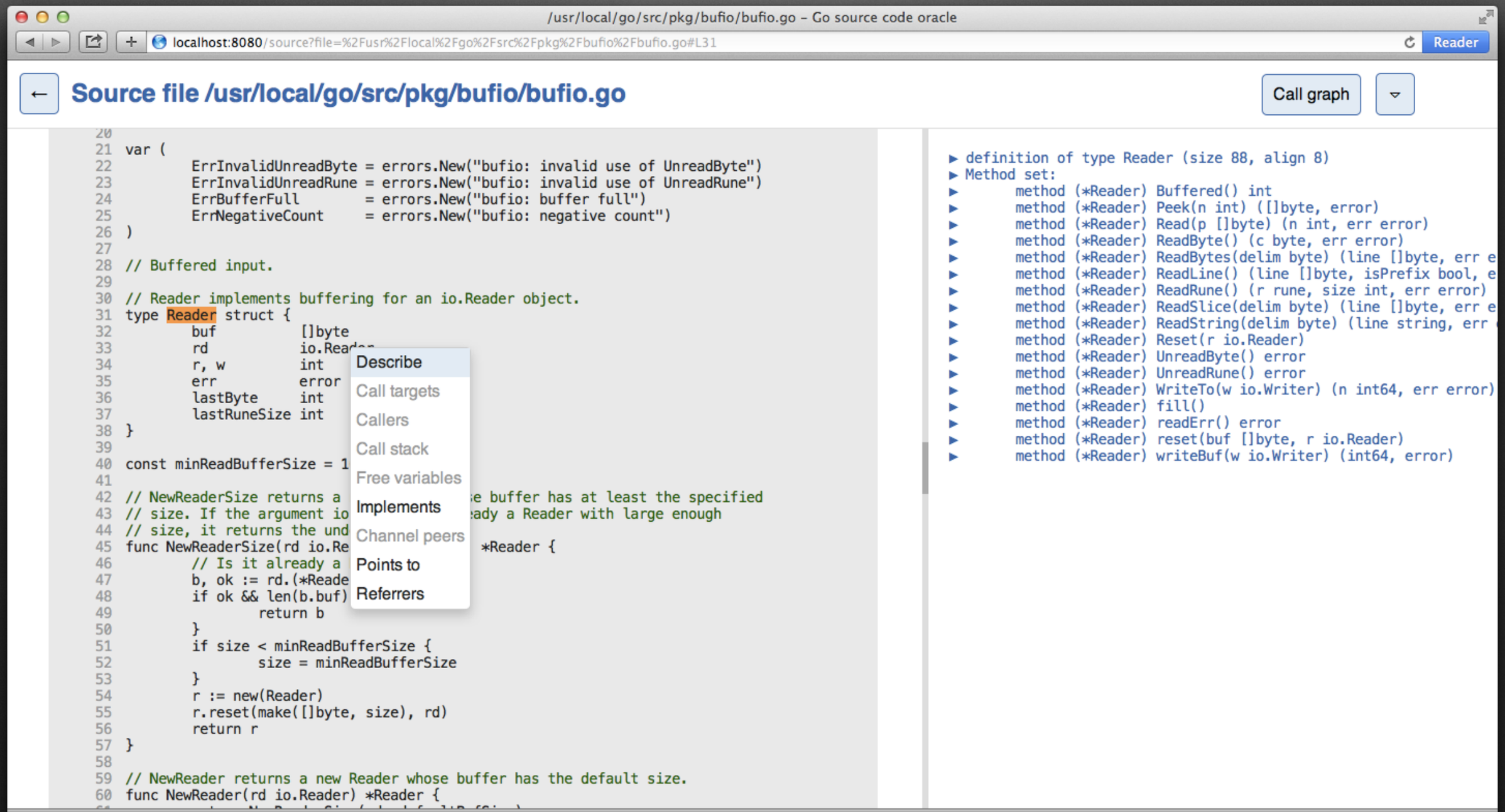


code.google.com/p/go.tools

“Yes, wonderful things” - Howard Carter, on first seeing Tutankhamun's tomb

go.tools/cmd/oracle answers questions about Go code

"using the oracle is as simple as selecting a region of source code, pressing a button, and receiving a precise answer"



github.com/fzipp/pythia

"Pythia is a web front-end for the Go source code oracle,
which is a source code comprehension tool for Go programs."

oracle library use includes:

- go/{token,scanner,ast,parser} parser
- .../go.tools/go/types type checker
- .../go.tools/go/importer package loader
- .../go.tools/go/ssa SSA IR
- .../go.tools/go/pointer pointer analysis
- .../go.tools/oracle oracle library

The 36 SSA (Single Static Assignment) Instructions

	Value?	Instruction?
*Alloc	✔	✔
*BinOp	✔	✔
*Builtin	✔	✔
*Call	✔	✔
*ChangeInterface	✔	✔
*ChangeType	✔	✔
*Convert	✔	✔
*DebugRef		✔
*Defer		✔
*Extract	✔	✔
*Field	✔	✔
*FieldAddr	✔	✔
*Go		✔
*If		✔
*Index	✔	✔
*IndexAddr	✔	✔
*Jump		✔
*Lookup	✔	✔
*MakeChan	✔	✔
*MakeClosure	✔	✔
*MakeInterface	✔	✔
*MakeMap	✔	✔
*MakeSlice	✔	✔
*MapUpdate		✔
*Next	✔	✔
*Panic		✔
*Phi	✔	✔
*Range	✔	✔
*Return		✔
*RunDefers		✔
*Select	✔	✔
*Send		✔
*Slice	✔	✔
*Store		✔
*TypeAssert	✔	✔
*UnOp	✔	✔

```
func fact(n int) int {
    if n == 0 {
        return 1
    }
    return n * fact(n-1)
}
```

go -> ssa example

```
# Name: main.fact
# Package: main
# Location: glug.go:9:6
func fact(n int) int:
.0.entry:                                                    P:0 S:2
   t0 = n == 0:int                                          bool
   if t0 goto 1.if.then else 2.if.done
.1.if.then:                                                  P:1 S:0
   return 1:int
.2.if.done:                                                  P:1 S:0
   t1 = n - 1:int                                            int
   t2 = fact(t1)                                             int
   t3 = n * t2                                               int
   return t3
```

Go SSA viewer

https://github.com/tmc/ssaview (https://github.com/tmc/ssaview)

Shows the SSA (Static Single Assignment) (http://en.wikipedia.org/wiki/Static_single_assignment_form) representation of input code.
Uses the wonderful go.tools/ssa (http://godoc.org/code.google.com/p/go.tools/ssa) package.

status: done

Input

```
1  package main            // example inspired by gobyexample.c
2  import  _ "runtime"     // required by go.tools/ssa/interp
3  func fact(n int) int {
4      if n == 0 {
5          return 1
6      }
7      return n * fact(n-1)
8  }
9  func main() { println("ten factorial is ",fact(10)) }
```

Result

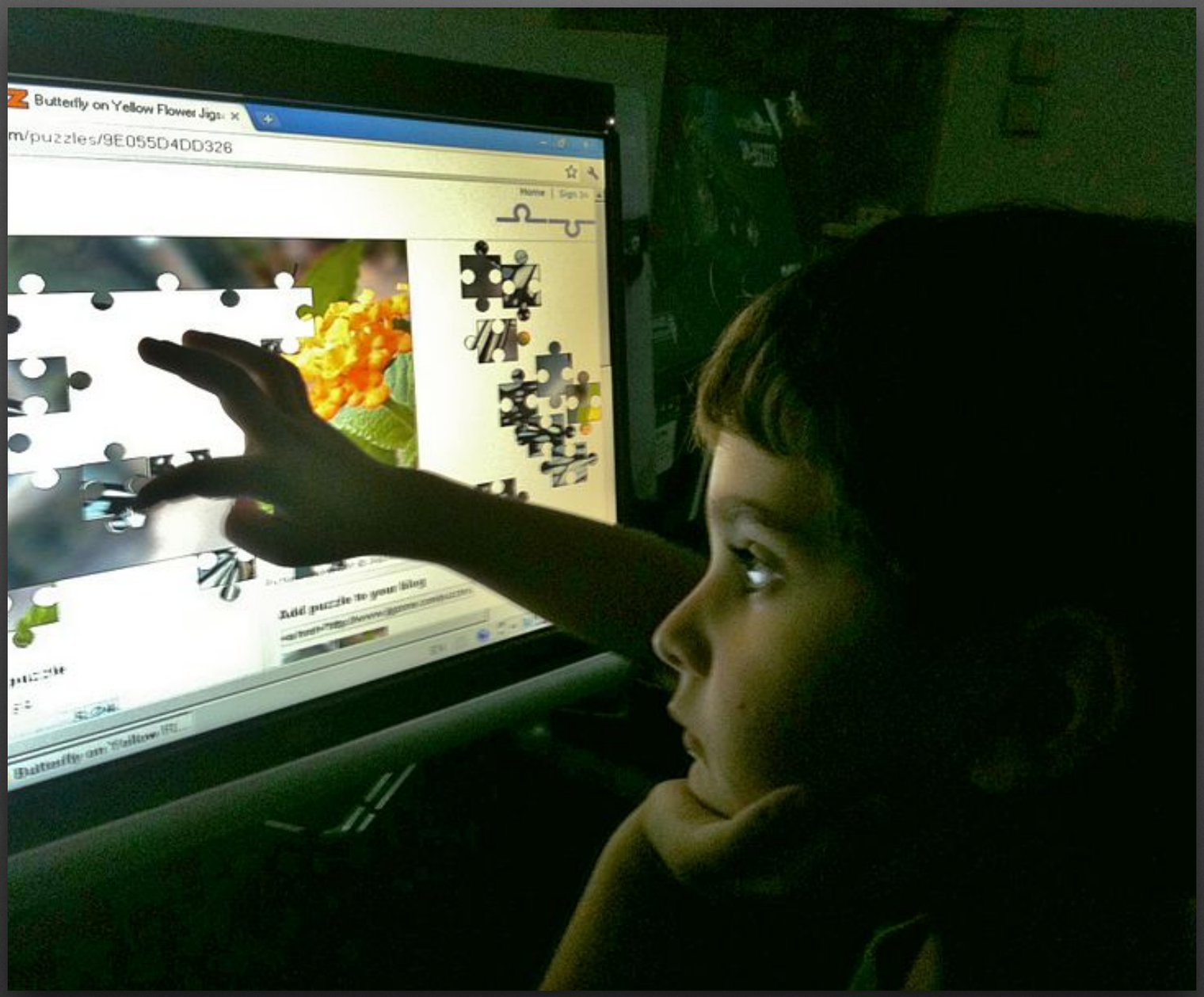
```
1  package main:
2    func  fact       func(n int) int
3    func  init       func()
4    var   init$guard bool
5    func  main       func()
6
7  # Name: main.init
8  # Package: main
9  # Synthetic: package initializer
10 func init():
11 .0.entry:
12     t0 = *init$guard
13     if t0 goto 2.init.done else 1.init.start
14 .1.init.start:
```

http://golang-ssaview.herokuapp.com/

To see this application running live

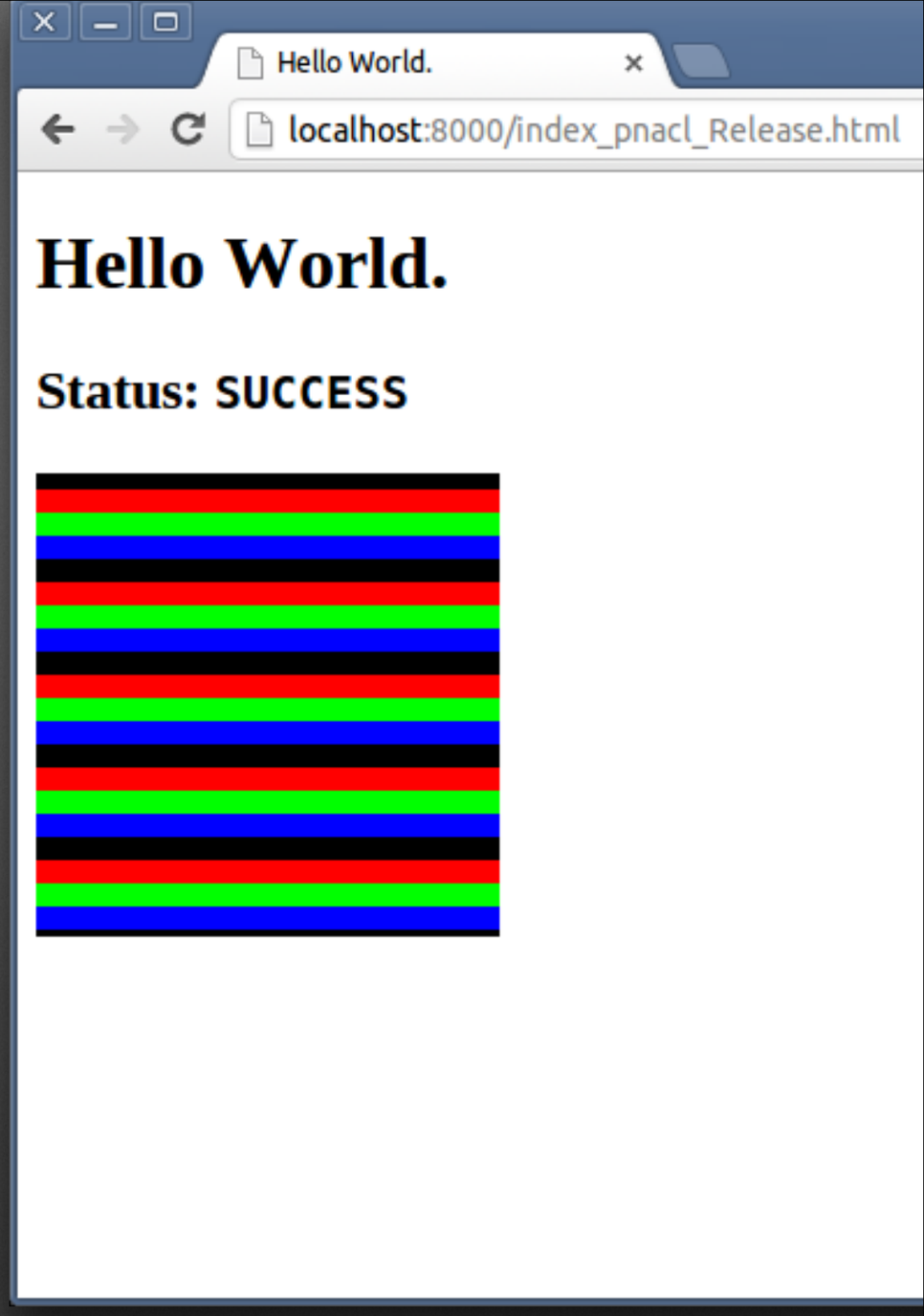
compilation sequences

<table>
<tr><th><u>TARDIS Go</u></th><th><u>llgo</u></th><th><u>GopherJS</u></th></tr>
<tr><td colspan="3">Go source => go/* libraries => AST</td></tr>
<tr><td colspan="3">AST => go.tools/go/types => types.Info</td></tr>
<tr><td colspan="2">AST + types.Info => go.tools/ssa => SSA</td><td rowspan="3">AST + types.Info => GopherJS => JavaScript</td></tr>
<tr><td>SSA => tardisgo => Haxe</td><td>SSA => llgo => LLVM module</td></tr>
<tr><td>Haxe => JavaScript, Flash, Java, C++, C#, PHP & Neko</td><td>LLVM modules => llgo-build => executable</td></tr>
</table>



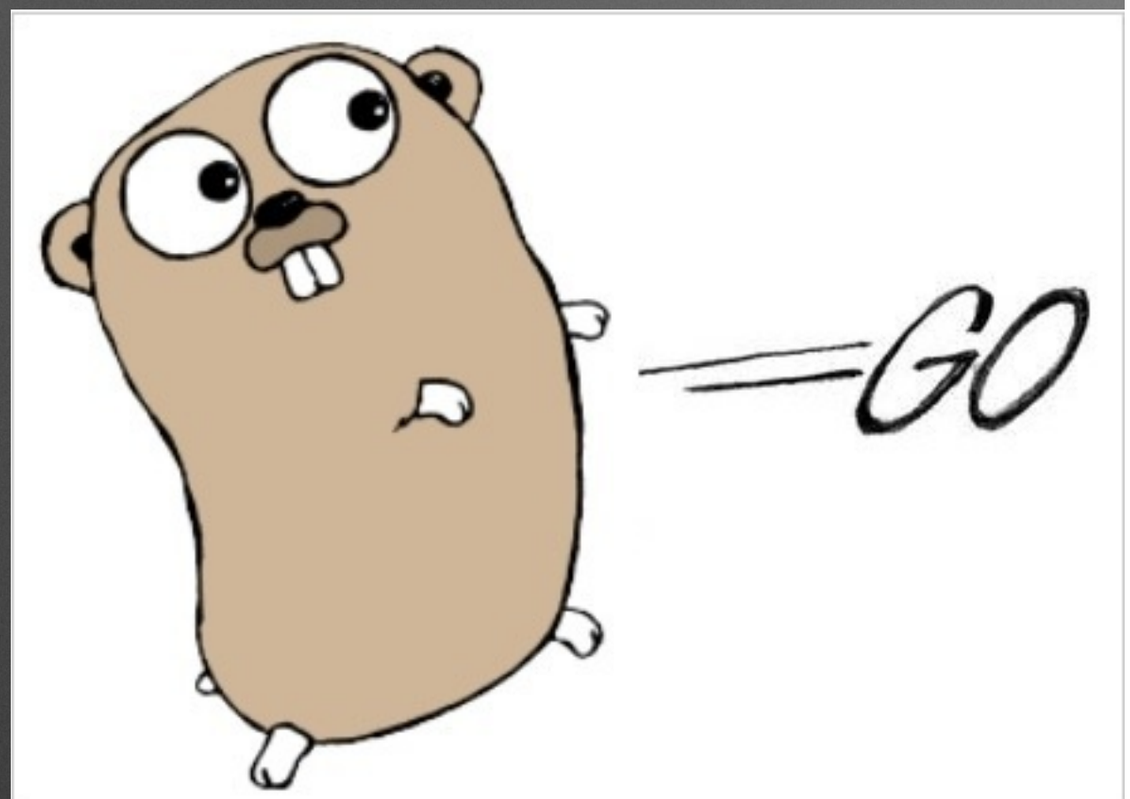
Prospects for client-side systems

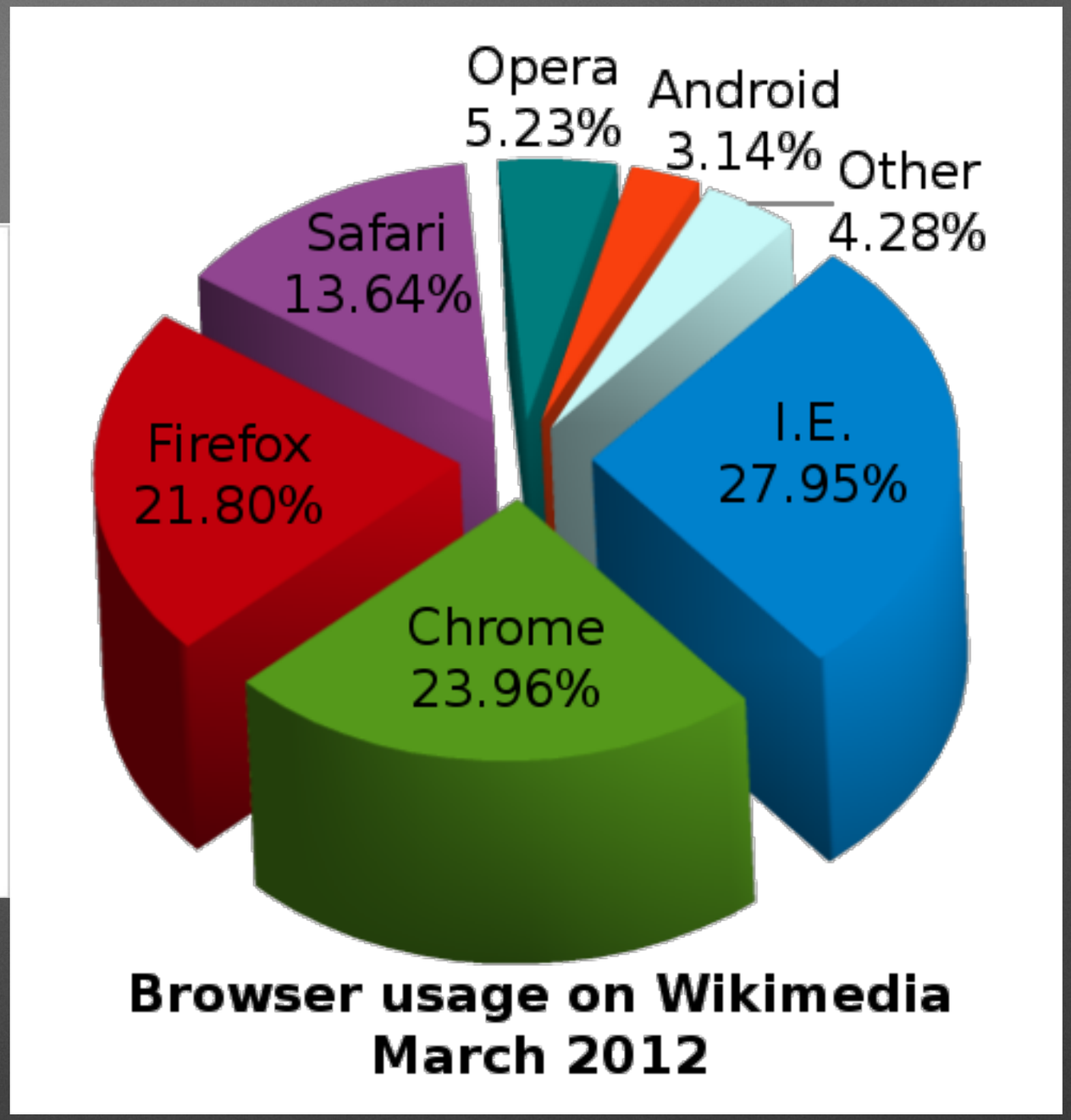
from the three “mutant” Go compilers.



github.com/axw/llgo

Go => extensive LLVM compiler infrastructure;
pNaCl or pepper.js + emscripten for applications in the browser?



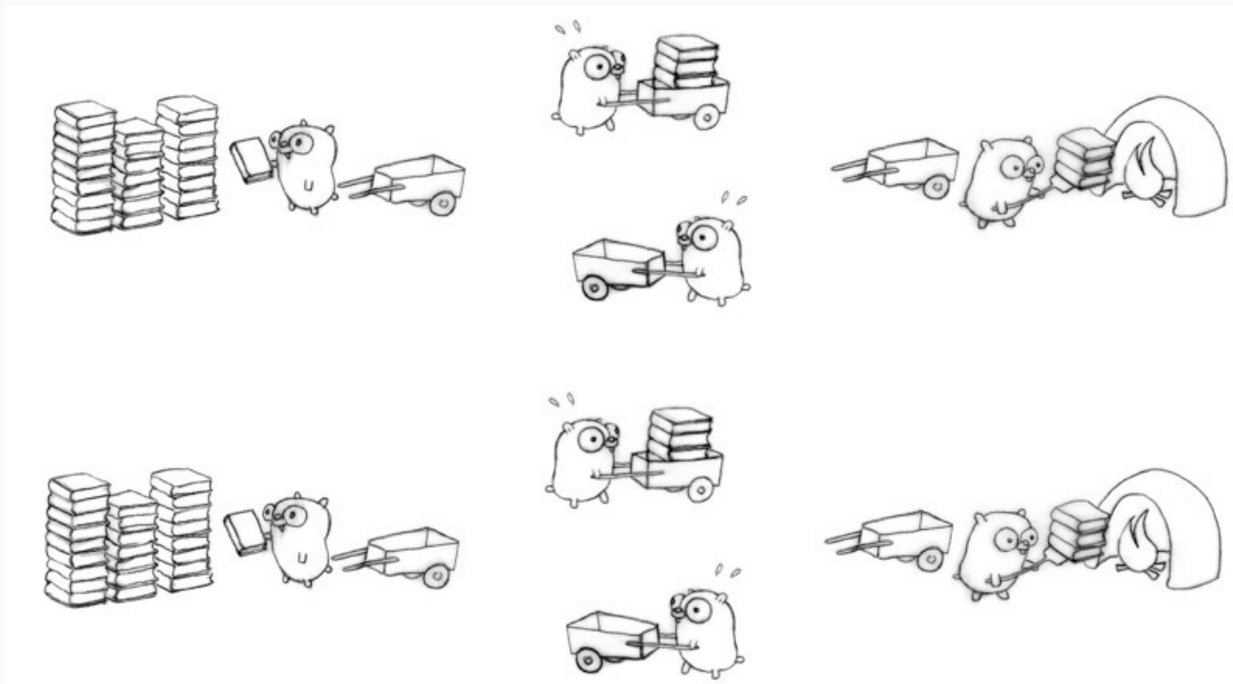


github.com/neelance/gopherjs

The opportunity to write front-end code in Go which will still run in all browsers.

Enj: HTML5 game engine for Go using GopherJS

Demonstration: http://ajhager.github.io/enj/



tardisgo.github.io

TARDIS Go: iOS, Android, Win, Mac, Linux, HTML5, Flash…

haxe.org

“Haxe can be compiled to all popular programming platforms with its fast compiler – JavaScript, Flash, NekoVM, PHP, C++, C# and Java – which means your apps will support all popular mobile devices, such as iOS, Android, BlackBerry and more.”

go -> ssa -> haxe

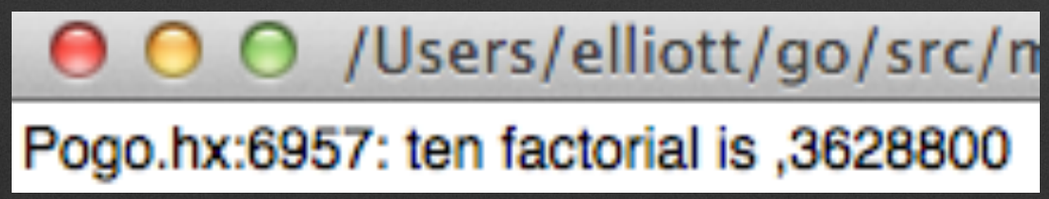
```
public function run():Pogo_main_fact {
while(true){
switch(_Next) {
case 0: // entry
this.setLatest(9,0);
this.SubFn0();
case 1: // if.then
this.setLatest(9,1);
_res= 1;
this._incomplete=false;
Scheduler.pop(this._goroutine);
return this; // return 1:int *ssa.Return @ glug.go:11:3
case 2: // if.done
this.setLatest(11,2);
this.SubFn1();
_SF1=Pogo_main_fact.call(this._goroutine,[],_t1);
_Next = -1;
return this;
case -1:
this.setLatest(13,-1);
_t2=_SF1.res();
 // _t2 = fact(t1) *ssa.Call @ glug.go:13:15
this.SubFn2();
_res= _t3;
this._incomplete=false;
Scheduler.pop(this._goroutine);
return this; // return t3 *ssa.Return @ glug.go:14:2
default: throw "Next?";}}}
private inline function SubFn0():Void {
var _t0:Bool;
_t0=(p_n==0); // _t0 = n == 0:int *ssa.BinOp @ glug.go:10:7
_Next=_t0 ? 1 : 2; // if t0 goto 1.if.then else 2.if.done *ssa.If near glug.go:10:7
}// end SubFn0
private inline function SubFn1():Void {
_t1=(p_n-1); // _t1 = n - 1:int *ssa.BinOp @ glug.go:13:17
}// end SubFn1
private inline function SubFn2():Void {
_t3=(p_n*_t2); // _t3 = n * t2 *ssa.BinOp @ glug.go:13:9
}// end SubFn2
```

```
func fact(n int) int {
    if n == 0 {
        return 1
    }
    return n * fact(n-1)
}
```

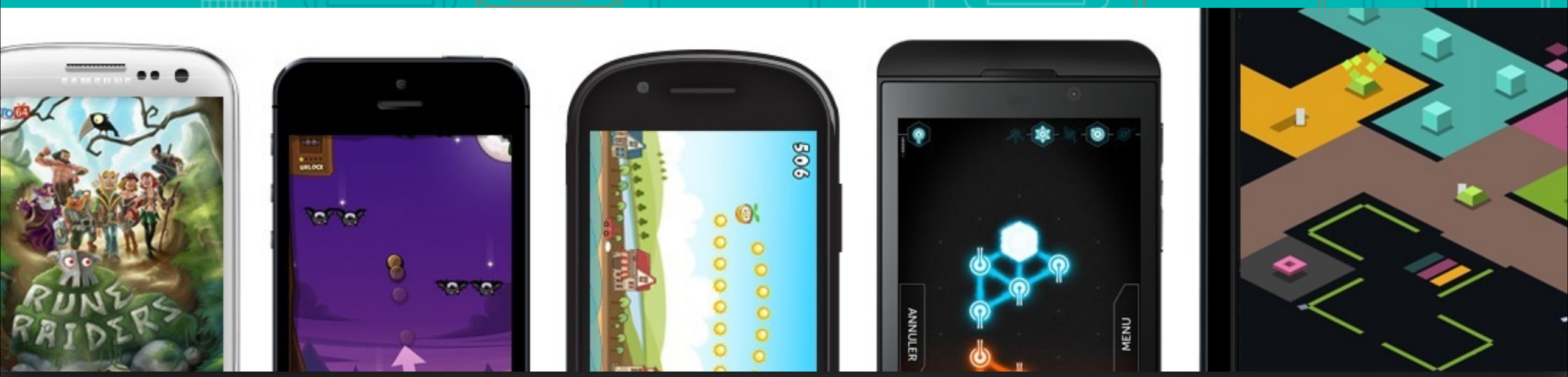
```
# Name: main.fact
# Package: main
# Location: glug.go:9:6
func fact(n int) int:
.0.entry:
   t0 = n == 0:int
   if t0 goto 1.if.then else 2.if.done
.1.if.then:
   return 1:int
.2.if.done:
   t1 = n - 1:int
   t2 = fact(t1)
   t3 = n * t2
   return t3
```

Go “fact(10)” running in JS, C++, Java, C#, PHP & Flash

```
Go->Haxe->JS (node<pogo.js):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->C++ (./cpp/Pogo):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->Java (java -jar java/java.jar):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->C# (mono ./cs/bin/cs.exe):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->PHP (php php/index.php):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->Flash (using flash player to test swf file):
```

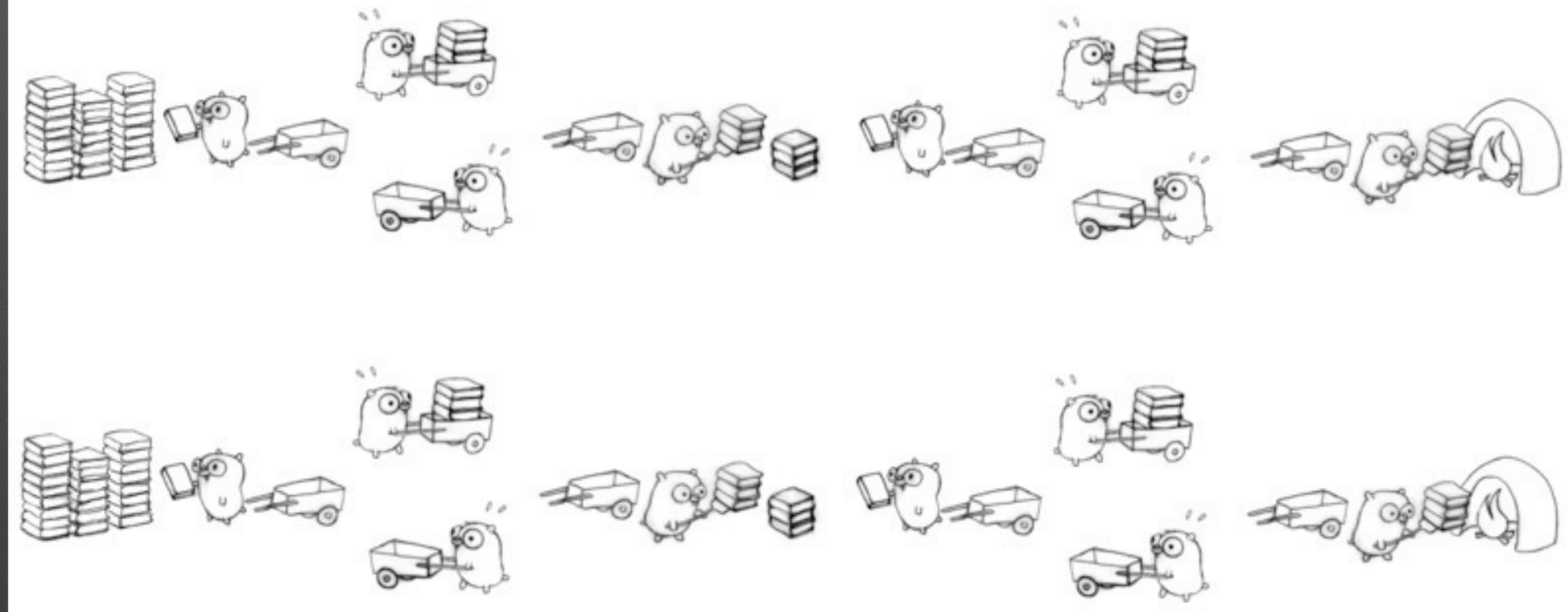


openFL



openfl.org

A Haxe based “cross-platform framework that targets Windows, Mac, Linux, iOS, Android, BlackBerry, Flash and HTML5” based on the Flash API



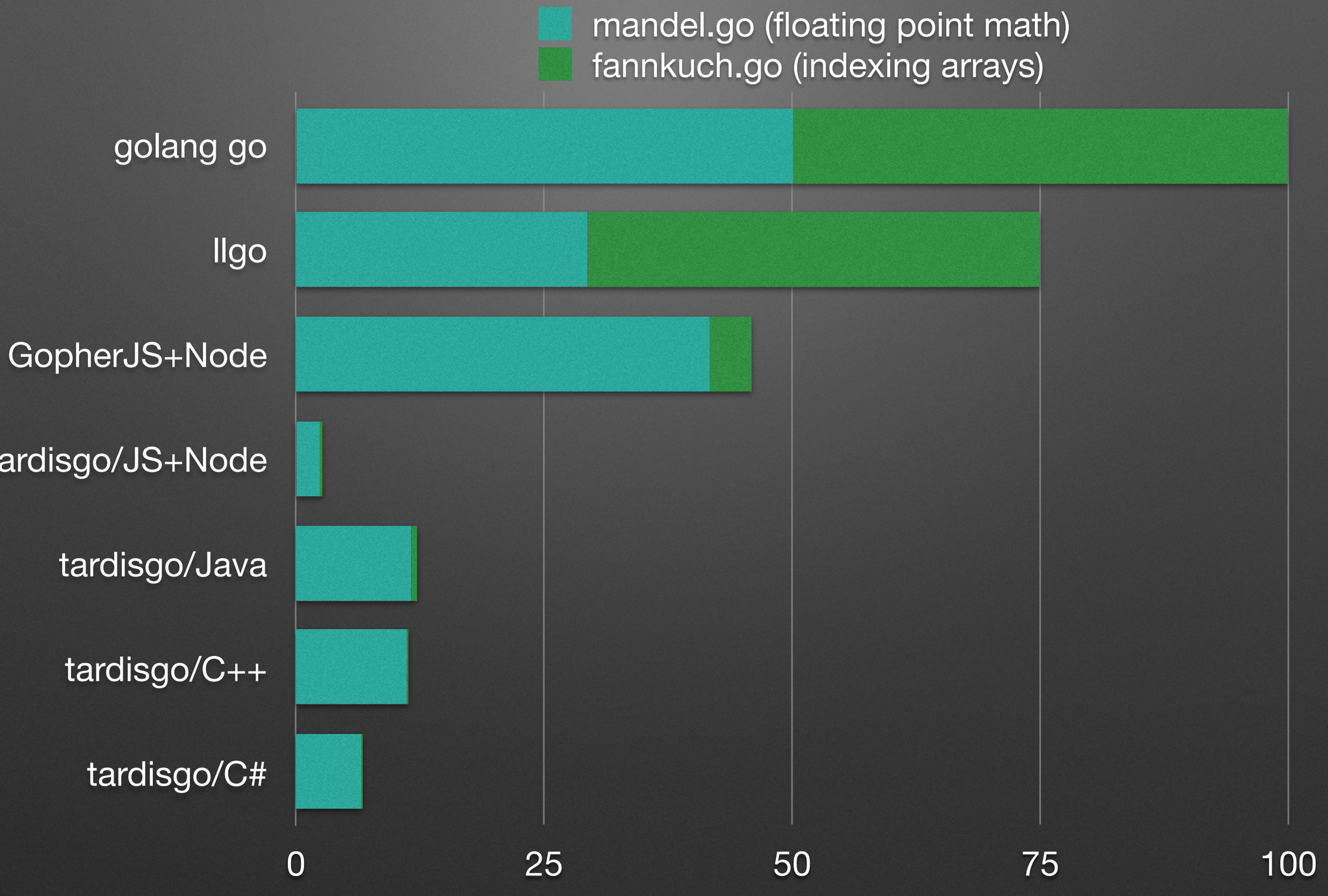
Concurrency is not Parallelism (it's better)

image from a talk by Rob Pike http://blog.golang.org/concurrency-is-not-parallelism
Demonstration: TARDIS Go animation of image at http://tardisgo.github.io

Premature optimization is the root of all evil (or at least most of it) in programming.

Donald Knuth

Go relative execution speeds snapshot, Jan 2014 implementation incompleteness => skewed results

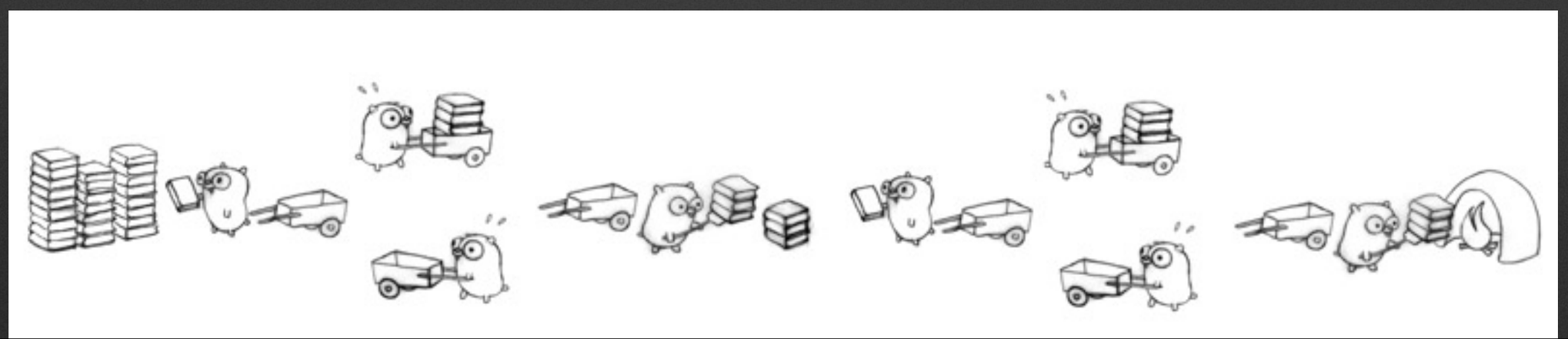


goroutines

- Use threads - great if the execution environment supports threads (golang), one per goroutine is simplest (llgo)

- Use co-routines, (if no threads) emulating multiple stacks, using SSA block # as the Program Counter (tardisgo)

- Don’t implement them - they add a significant execution overhead and are not always required (GopherJS at present, Tulgo)



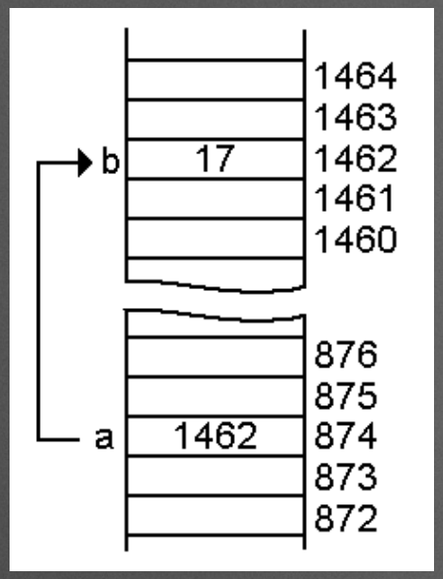
type issues

- float32 and complex64 currently modelled as float64 and complex128

- int8, int16, uint8 and uint16 each need additional processing to mask out the high-order bits and (maybe) extend the sign as part of each computation

- int64 and uint64 require emulation on platforms like JS without 64-bit arithmetic

- string might be implemented as utf-8 or utf-16



memory and pointers

- If pointers don’t exist and must be emulated, the simplest option is native simulation [emscripten.org] and pointers are integers…
- …but Go implementations must do their own garbage collection, leading to potential problems when interacting with the host environment
- However, unlike C, Go pointers can’t reference outside their type, so memory can be a collection of host objects, with garbage collection by the host

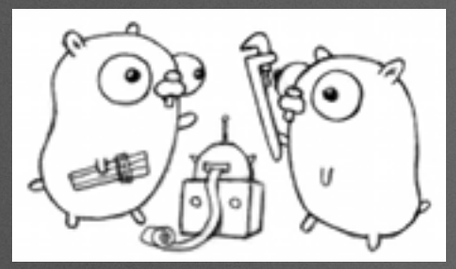
pointer emulation methods

- tardisgo (at present): pointers are offsets into heap objects which are arrays of dynamic objects, which may each contain other arrays of dynamic objects and so on ad infinitum (must work for all 7 targets)

- GopherJS: pointers are objects that are individually created to implement the type required, with getter and setter functions; except for structures, where they directly use the JS object instance as a struct pointer, without the overhead of an additional wrapper

unsafe pointers

- native (llgo)
- simulated Richard Musiol wrote: *‘That's a hairy topic. Yes, GopherJS has some support for them, at least as far as required for making the "syscall" package work and some other special cases. Heavy pointer arithmetic like "encoding/gob" tries to do is not supported.’*
- not supported (tardisgo at present)



Go standard library

- Target runtime functions are used in many places => a matching set is required in Go or the target language
- Large code size for client-side use, or for some targets e.g. unicode.init()
- Some compute-intensive packages may be inappropriate for an emulated environment
- Some packages may call system functions unavailable in the target environment

GopherJS & JS libraries

- A special "js" package gives access to raw JS objects via methods and struct field tags, allowing access to the DOM or jQuery etc. :

```
document := js.Global("document")

view := document.Call("createElement", "canvas")

target := document.Call("getElementById", container)

if target.IsNull() {

      target = document.Get("body")

}

target.Call("appendChild", view)
```

TARDIS Go & Haxe libraries

- Go packages beginning with an underscore define Haxe functionality; they can be auto-generated from the Haxe code or library documentation with target-specific capital letter prefixes:

```
s := string(_haxeapi.Xhaxe_Http_requestUrl(someURL))
```

```
switch tardisgolib.Platform() {

case "cpp": _haxeapi.Pcpp_Lib_println(s)

case "php": _haxeapi.Hphp_Lib_println(s)

default: println(s)

}
```

review

- Go’s specification
- go.tools repository
- “mutant” compilers
- design decisions

Write your own Go compiler ...look at an existing project!

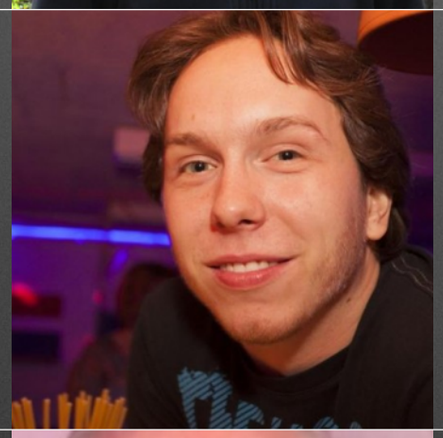
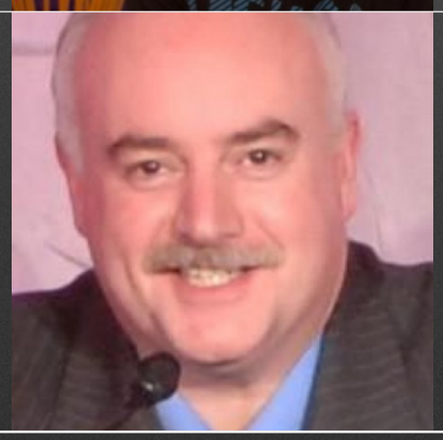
llgo		github.com/axw/llgo
GopherJS		github.com/neelance/gopherjs
tardisgo		tardisgo.github.io

Image Sources

- Gopher Logo by Renée French
- Gopher image rendered in blender by David Ackerman
- http://commons.wikimedia.org/
- http://www.wikipedia.org/
- https://www.google.com/doodles/howard-carters-138th-birthday
- Project related images from relevant project sites
- Other images self-created