netcetera

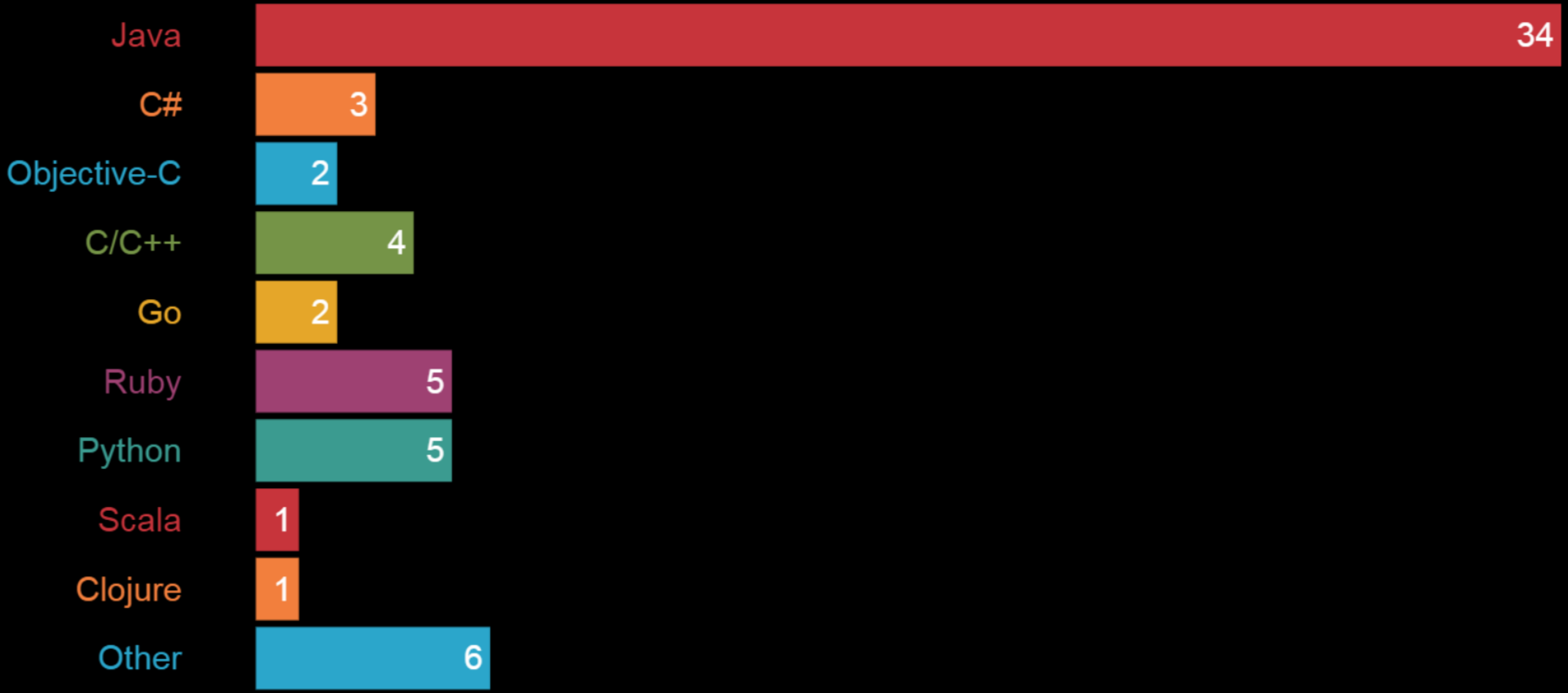
Go or no go

Beyond Java: Go for Java developers

Andrej Vckovski <andrej.vckovski@netcetera.com>

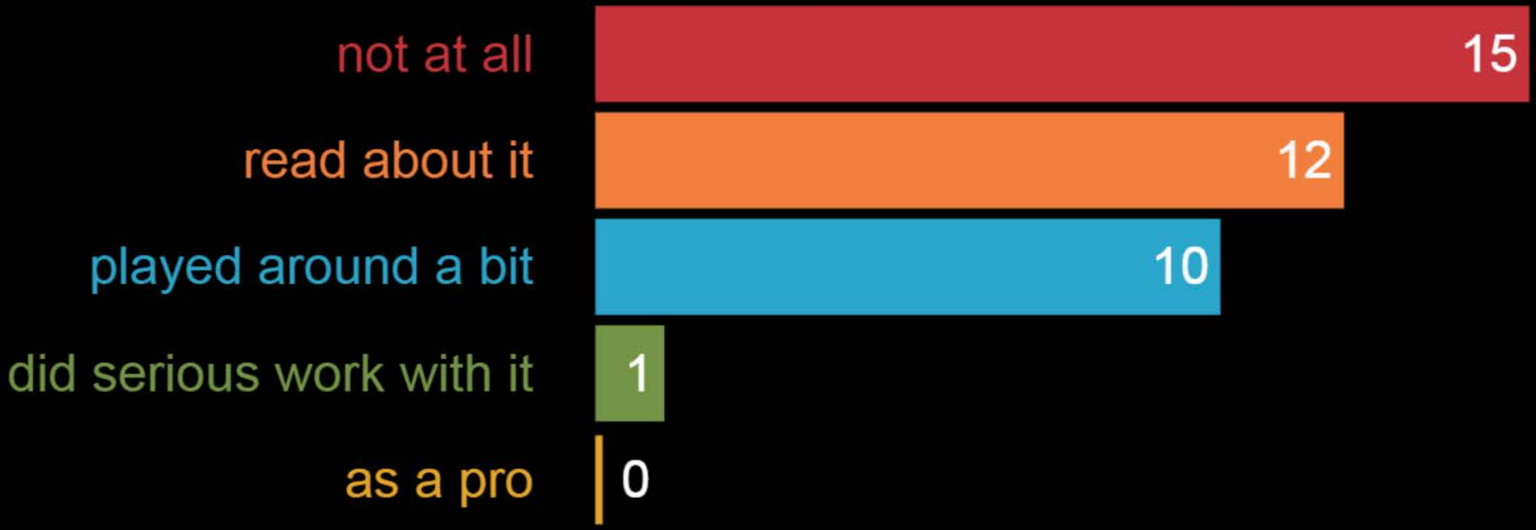
Before we start …

My current development context



Total answers: 63

I have used Go



Total answers: 38

Direct Poll

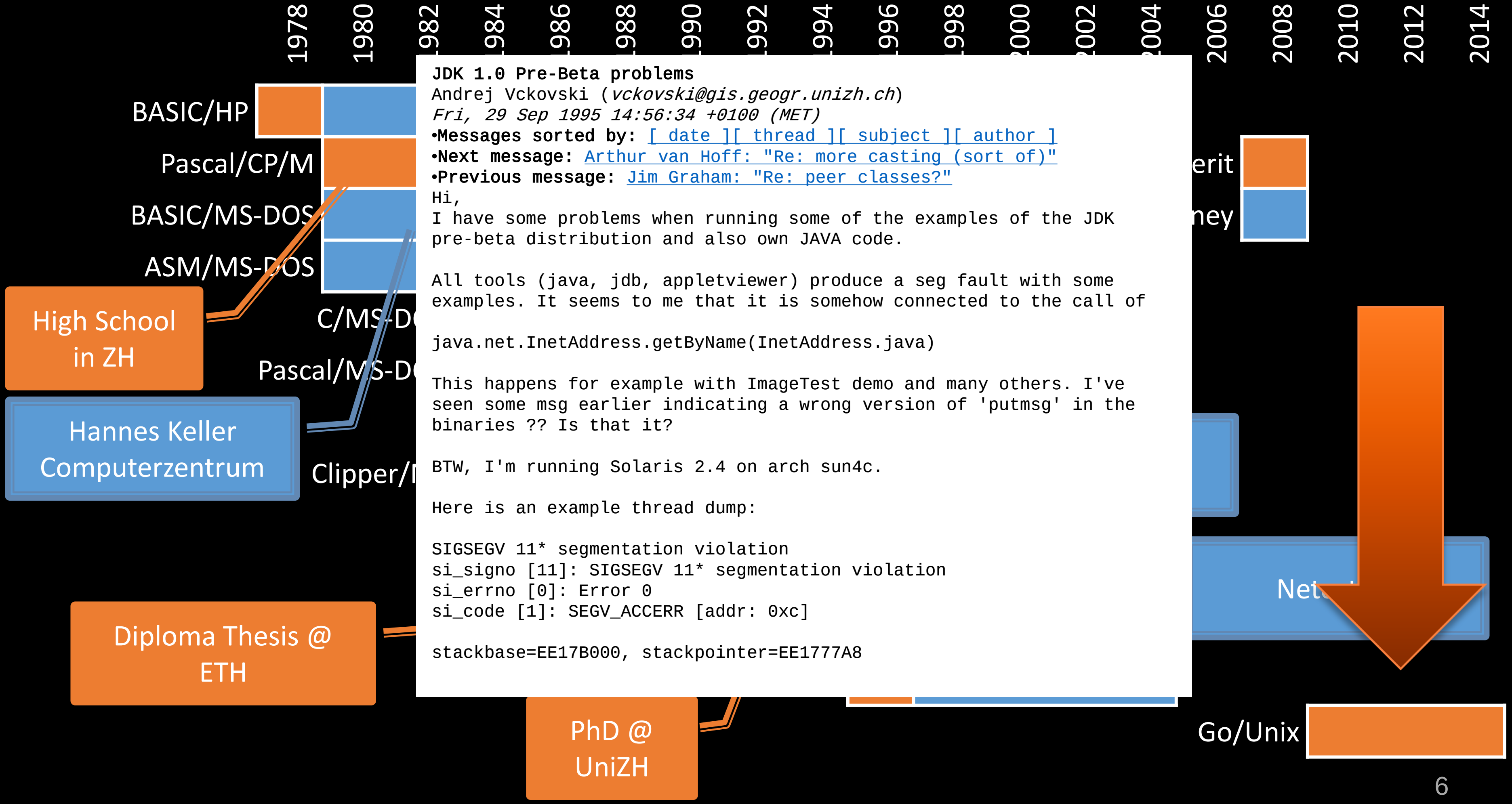
Motivaton

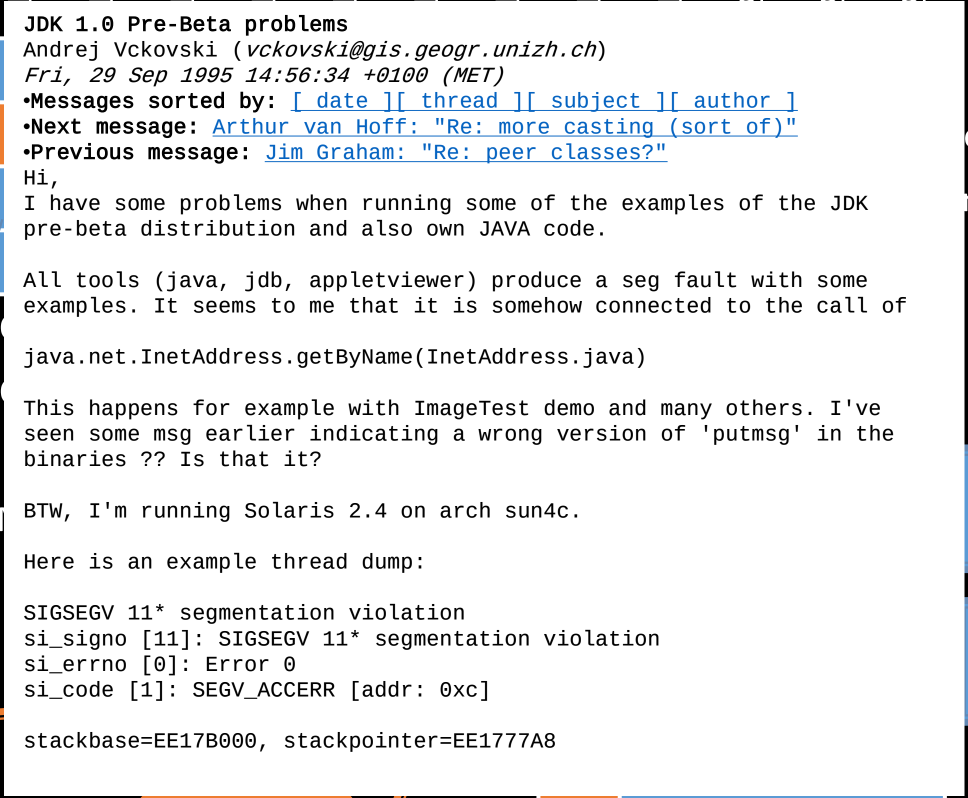
- About me, us and them
- The showcase

Go programming

- The language
- Running go programs
- Tool-chain
- Ecosystem

So What

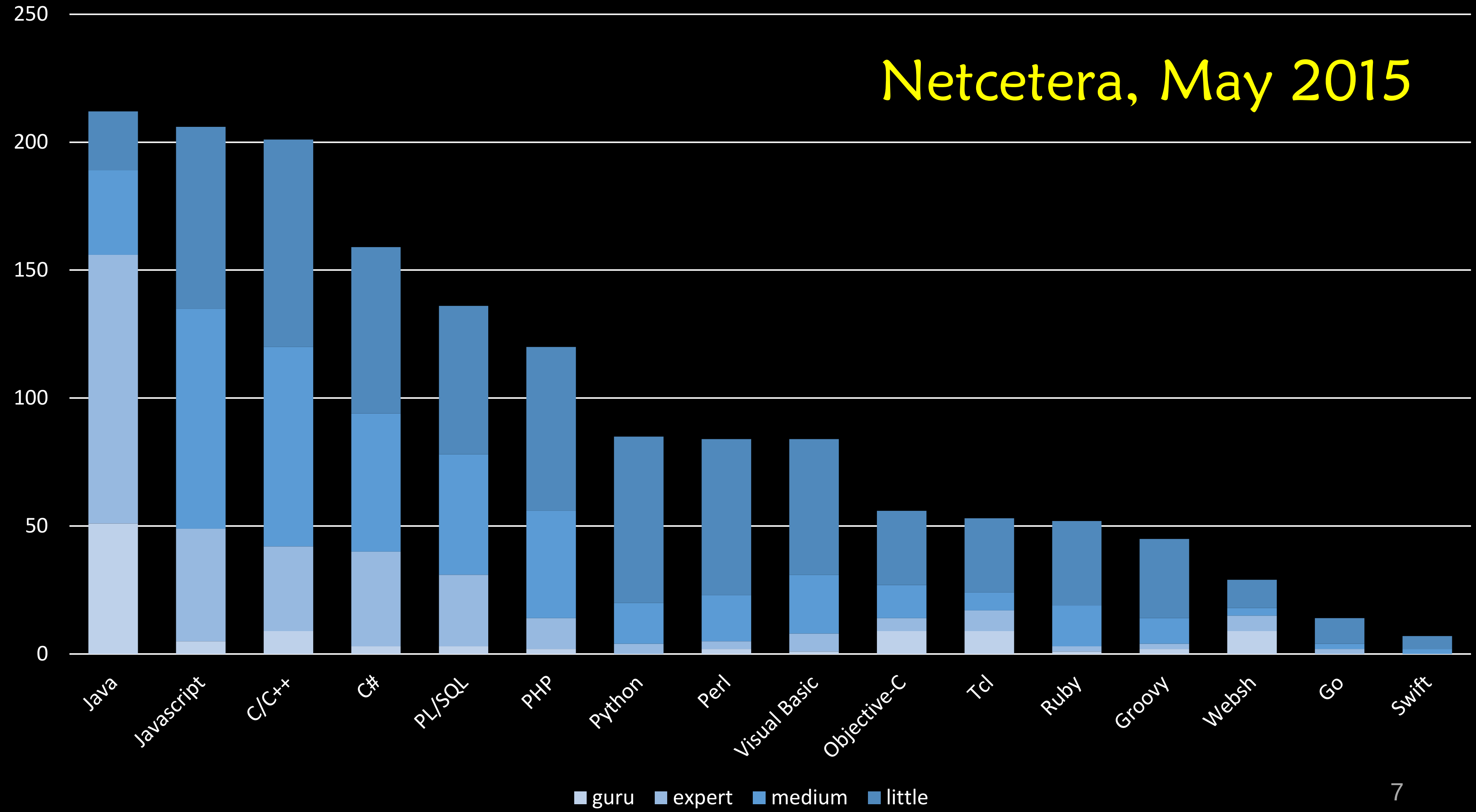


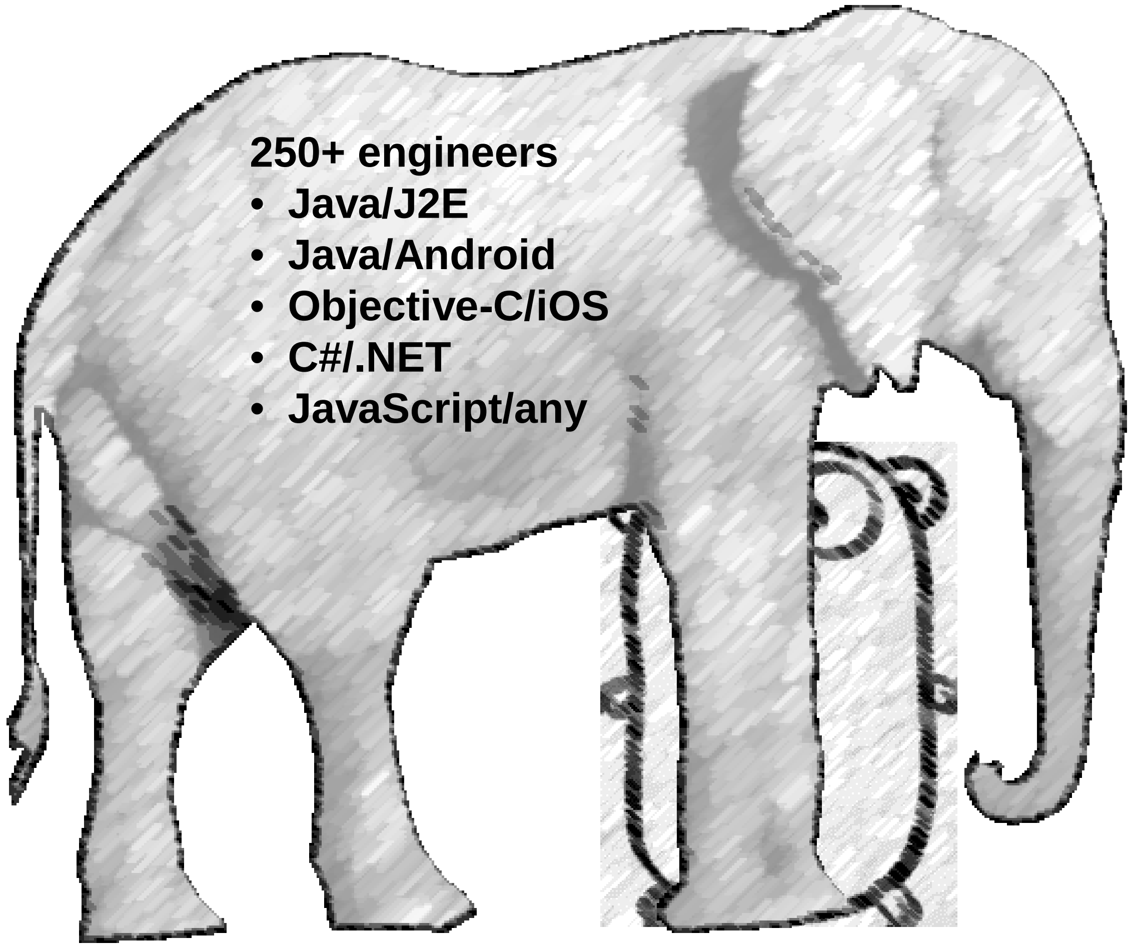


JDK 1.0 Pre-Beta problems
Andrej Vckovski (*vckovski@gis.geogr.unizh.ch*)
Fri, 29 Sep 1995 14:56:34 +0100 (MET)
•**Messages sorted by:** [date][thread][subject][author]
•**Next message:** Arthur van Hoff: "Re: more casting (sort of)"
•**Previous message:** Jim Graham: "Re: peer classes?"

```
Hi,
I have some problems when running some of the examples of the JDK
pre-beta distribution and also own JAVA code.

All tools (java, jdb, appletviewer) produce a seg fault with some
examples. It seems to me that it is somehow connected to the call of

java.net.InetAddress.getByName(InetAddress.java)

This happens for example with ImageTest demo and many others. I've
seen some msg earlier indicating a wrong version of 'putmsg' in the
binaries ?? Is that it?

BTW, I'm running Solaris 2.4 on arch sun4c.

Here is an example thread dump:

SIGSEGV 11* segmentation violation
si_signo [11]: SIGSEGV 11* segmentation violation
si_errno [0]: Error 0
si_code [1]: SEGV_ACCERR [addr: 0xc]

stackbase=EE17B000, stackpointer=EE1777A8
```

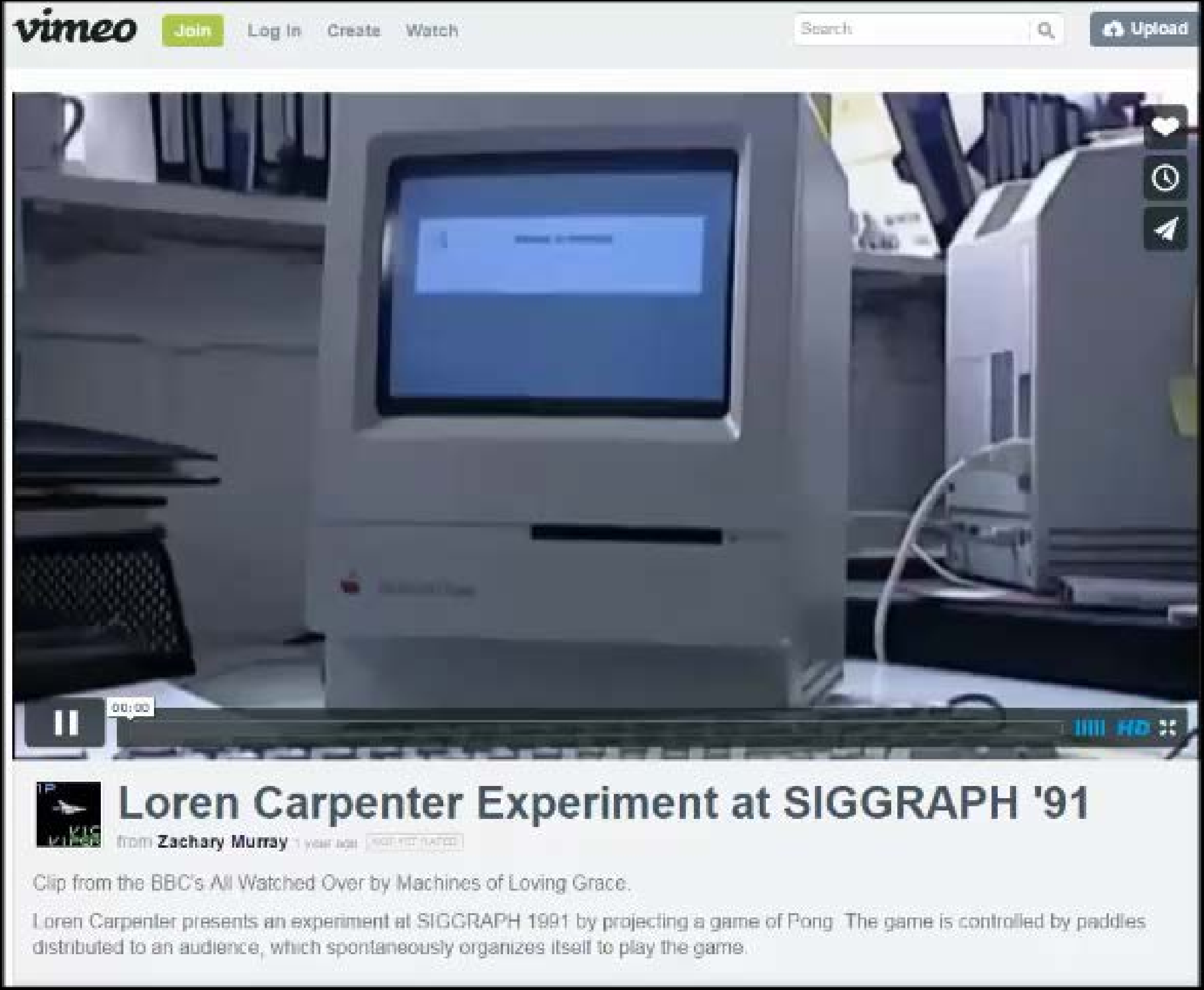
Netcetera, May 2015





250+ engineers

- **Java/J2E**
- **Java/Android**
- **Objective-C/iOS**
- **C#/.NET**
- **JavaScript/any**

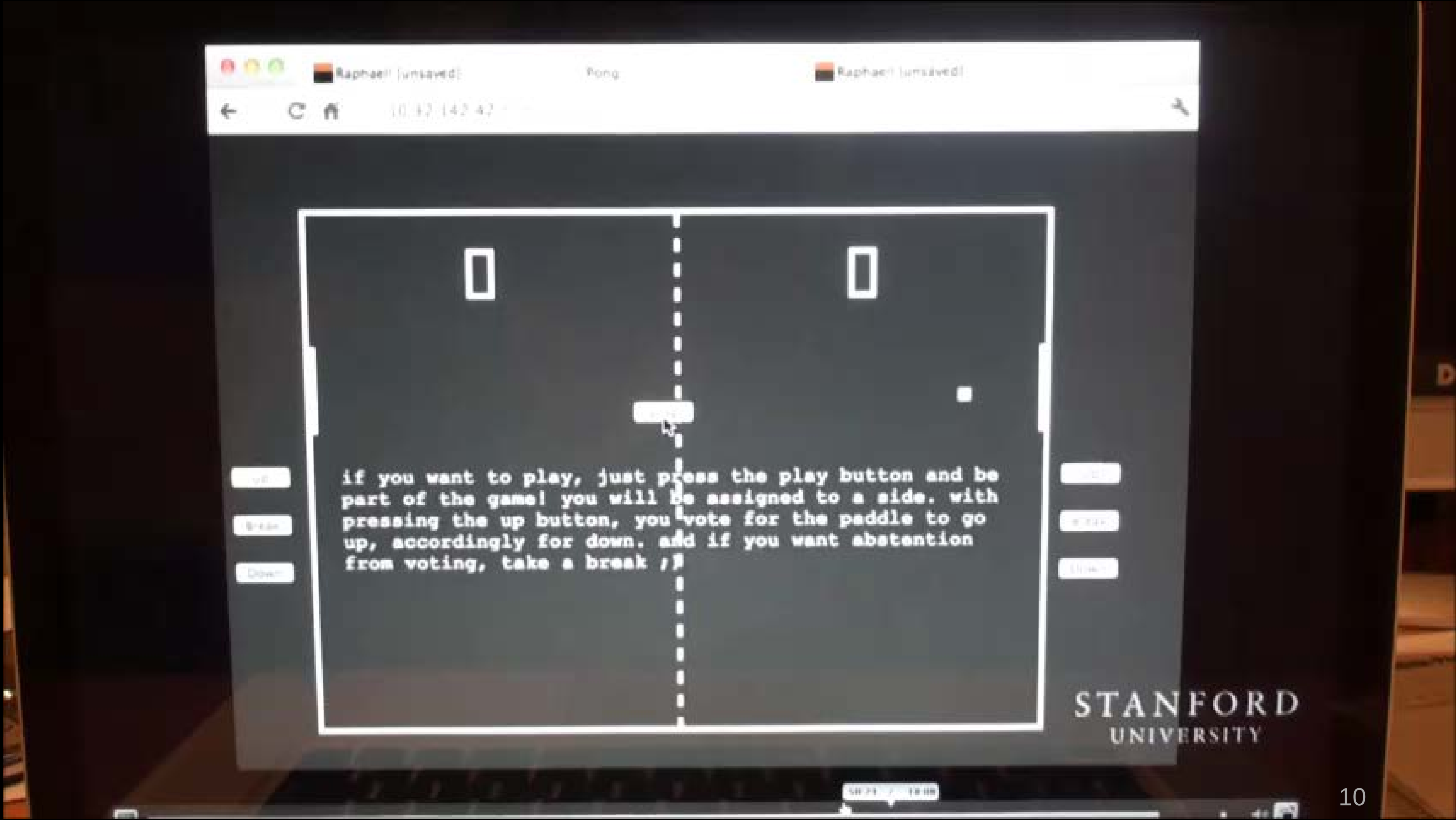


Loren Carpenter Experiment at SIGGRAPH '91

from **Zachary Murray** 1 year ago

Clip from the BBC's All Watched Over by Machines of Loving Grace.

Loren Carpenter presents an experiment at SIGGRAPH 1991 by projecting a game of Pong. The game is controlled by paddles distributed to an audience, which spontaneously organizes itself to play the game.



if you want to play, just press the play button and be part of the game! you will be assigned to a side. with pressing the up button, you vote for the paddle to go up, accordingly for down. and if you want abstention from voting, take a break ;)

Direct Poll

The application:
Very Instant Massive (Audience) Polling

- Presentations like this one
- TV shows with added interactivity
- Pause entertainment in a stadium
- Flipped Classrooms

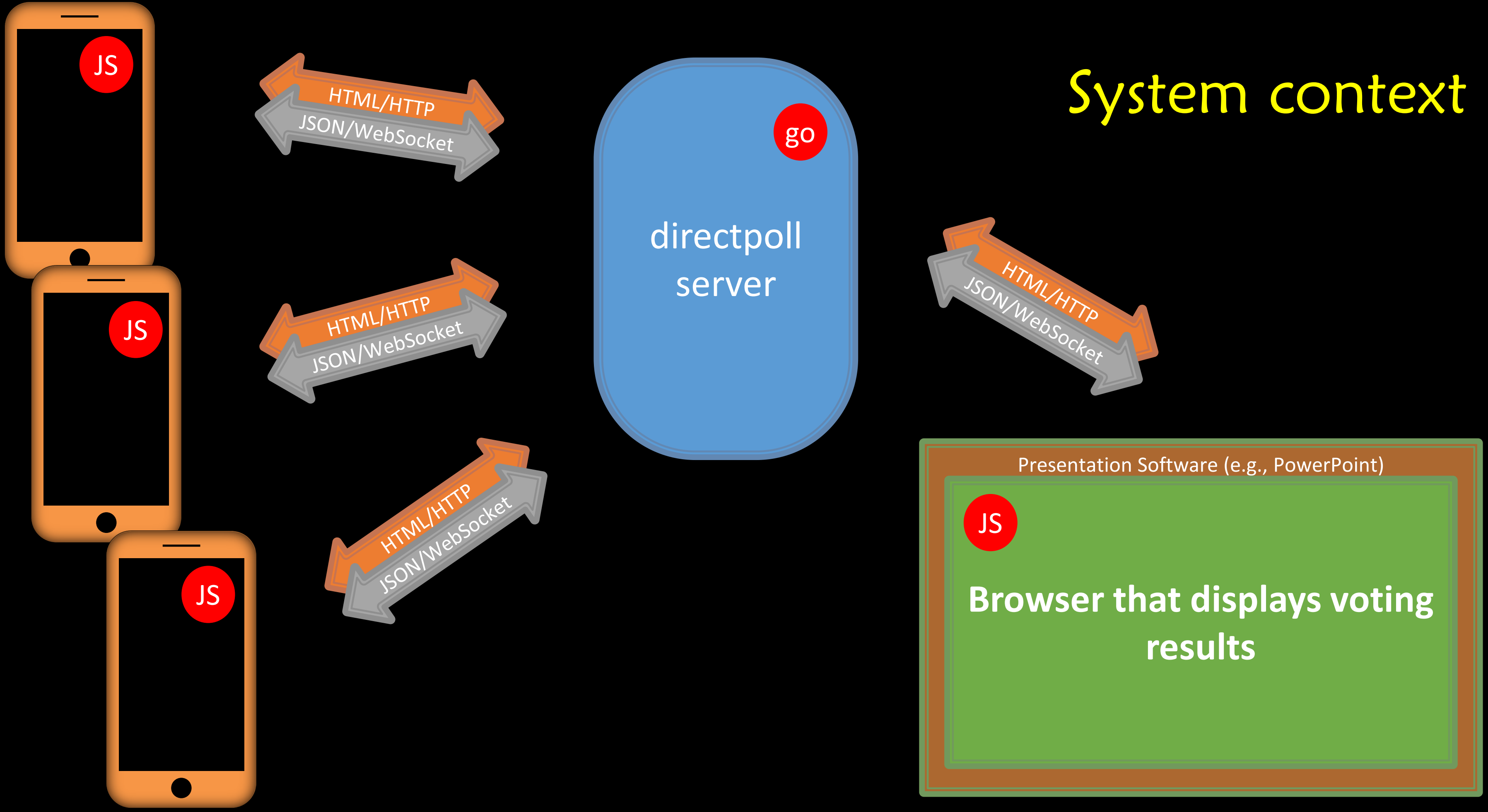
© Nhenze

© Chris Lawrence

© Loozrboy

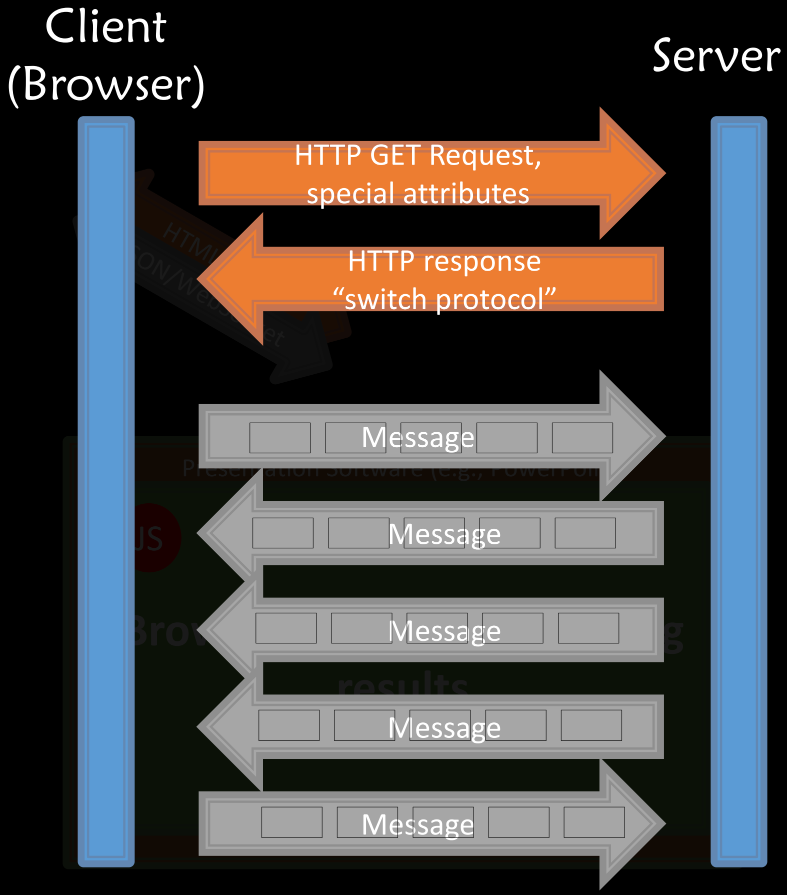
© Jeff Chenqinyi

System context

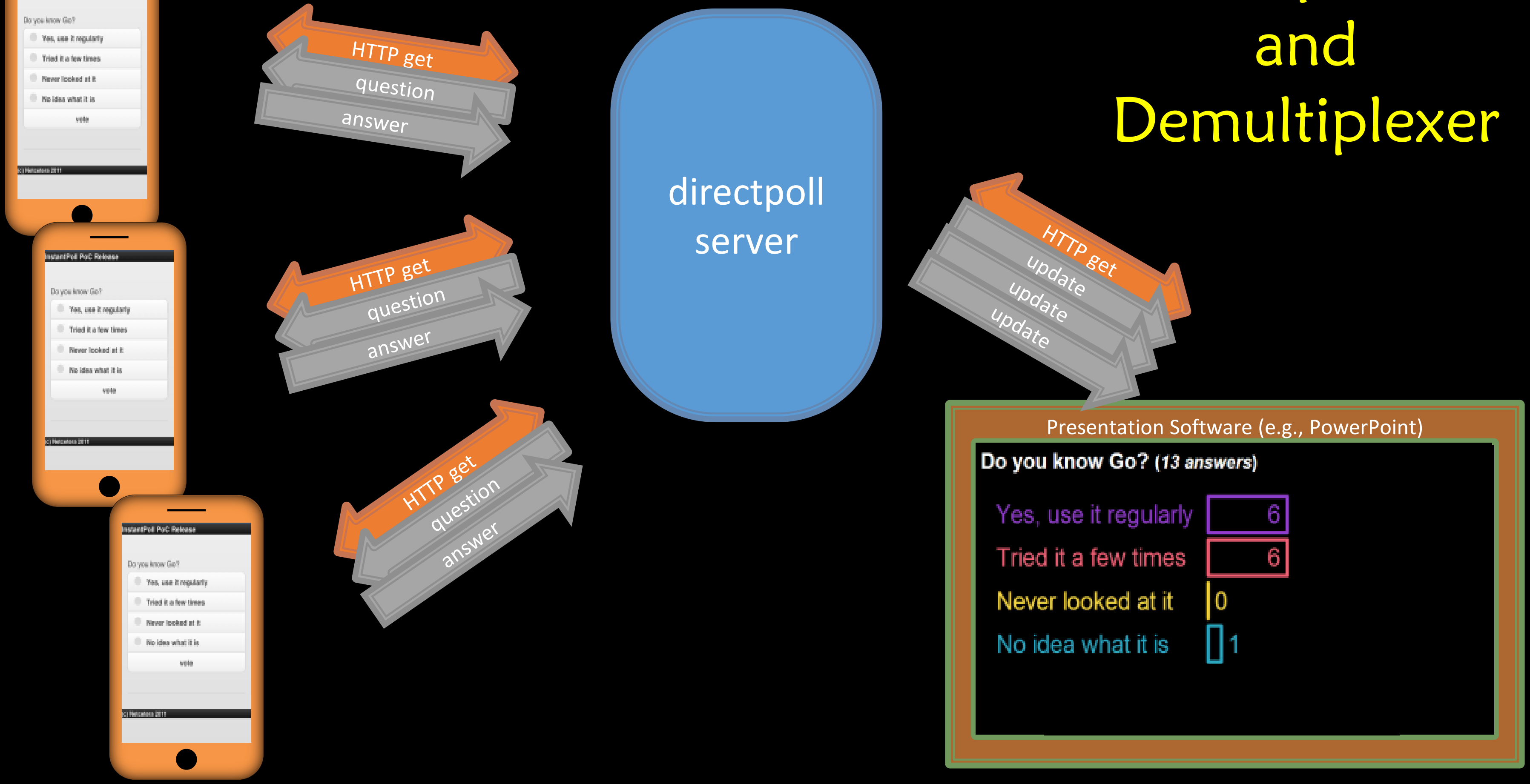


WebSockets

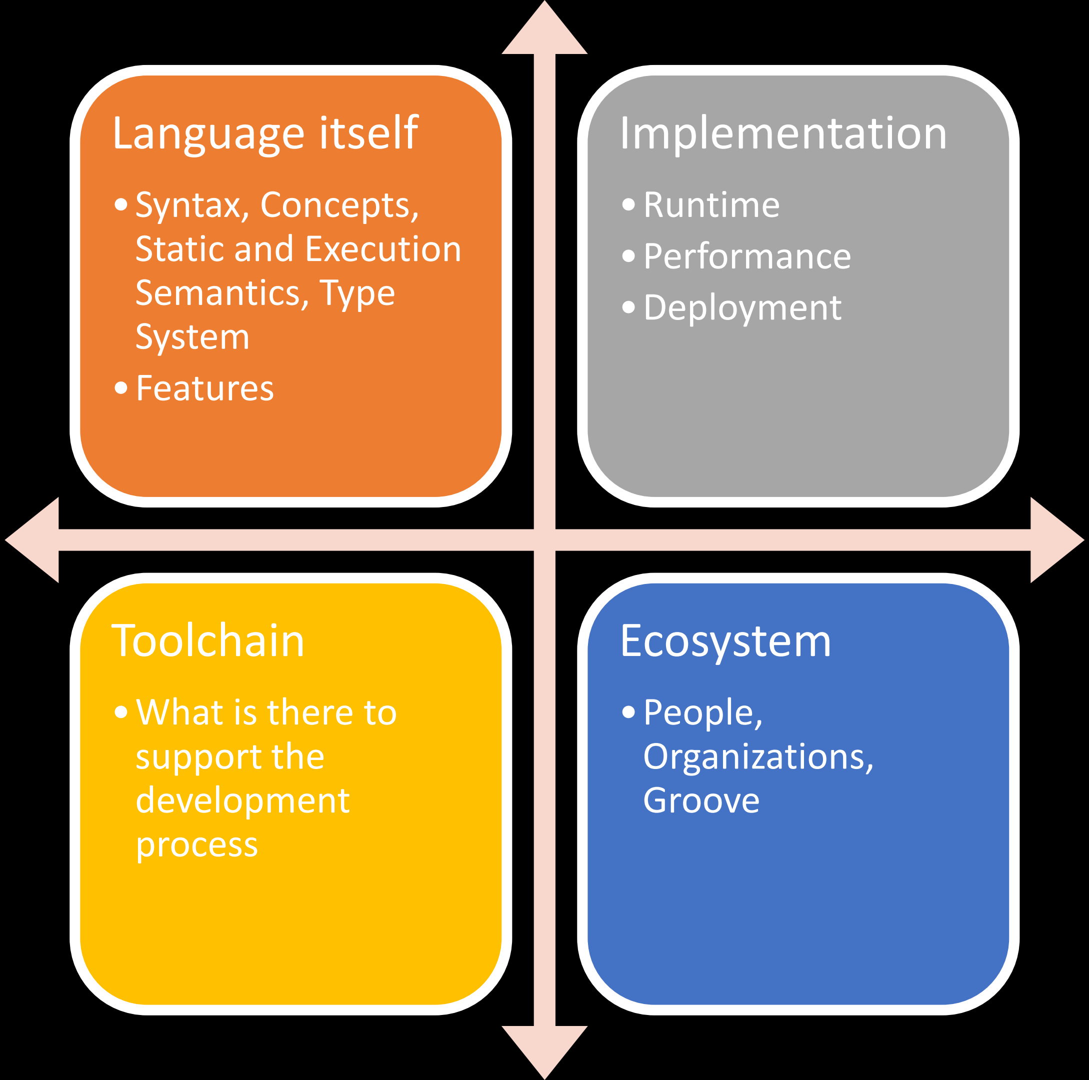
- Full-duplex conversation over TCP connection
- RFC 6455
- Available in most modern browsers
- Simple JavaScript binding
- Handshake by HTTP, then user-defined messages over the same socket


Client (Browser)
Server
HTTP GET Request, special attributes
HTTP response "switch protocol"
Message
Message
Message
Message
Message

Multiplexer and Demultiplexer



The Language



Language itself

- Syntax, Concepts, Static and Execution Semantics, Type System
- Features

Implementation

- Runtime
- Performance
- Deployment

Toolchain

- What is there to support the development process

Ecosystem

- People, Organizations, Groove

About Go

"Software is getting slower more rapidly than hardware becomes faster."
(N.W., 1995)

Credits

Content taken partly from
"Go for Java Programmers" by Sameer Ajmani

A talk presented at NYJavaSIG (http://javasig.com) on April 23, 2015.

Watch the talk on YouTube (https://www.youtube.com/watch?v=_c_tQ6_3cCg)

What is Go?

"Go is an open source programming language that makes it easy to build simple, reliable, and efficient software."

golang.org (http://golang.org)

History

2007: Design started

- Robert Griesemer, Rob Pike, and Ken Thompson.
- Ian Lance Taylor and Russ Cox.

2009: Open source

2012: Go 1

Design drivers

- Robustness
- Concurrency
- Performance
- Large codebases
- Fast development cycles
- Easy to learn
- No bagagge

Go and Java have much in common

- C family (imperative, braces)
- Statically typed
- Garbage collected
- Memory safe (nil references, runtime bounds checks)
- Variables are always initialized (zero/nil/false)
- Methods
- Interfaces
- Type assertions (`instanceof`)
- Reflection

Go differs from Java in several ways

- Programs compile to machine code. There's no VM.
- Statically linked binaries
- Function values and lexical closures
- Built-in strings (UTF-8)
- Built-in generic maps and arrays/slices
- Built-in concurrency

Go intentionally leaves out many things

- No classes
- No constructors
- No inheritance
- No `final`
- **No exceptions**
- No annotations
- **No generics**

Interesting details

- Functions and methods can return multiple values
- There is a blank identifier (_), no 'dummy' anymore
- No casting, explicit conversions needed, e.g., from one int32 to int16
- simple declaration and initialization: `foo := 3`
- Compiler is very picky: Unused imports and variables are not tolerated
- Starting with an Uppercase or lowercase letter defines visibility of methods and fields of a package
- Very complete standard library, especially w.r.t. to networking stuff
- "Duck typing"
- `defer` to define stuff that should happen a function exit

Examples

Hello, world!

Main.java

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

hello.go

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello, JUGS")
}
```

Run

Hello, web server

```
package main

import (
    "fmt"
    "log"
    "net/http"
)

func main() {
    http.HandleFunc("/hello", handleHello)
    fmt.Println("serving on http://localhost:7777/hello")
    log.Fatal(http.ListenAndServe("localhost:7777", nil))
}

func handleHello(w http.ResponseWriter, req *http.Request) {
    log.Println("serving", req.URL)
    fmt.Fprintln(w, "Hello, 世界!")
}
```

Run

Concurrency

Communicating Sequential Processes (Hoare, 1978)

Concurrent programs are structured as independent processes that execute sequentially and communicate by passing messages.

"Don't communicate by sharing memory, share memory by communicating."

Go primitives: goroutines, channels, and `select` statement.

Goroutines

Goroutines are like lightweight threads.

They start with tiny stacks and resize as needed.

Go programs can have hundreds of thousands of them.

Start a goroutine using the go statement:

```
go f(args)
```

The Go runtime schedules goroutines onto OS threads.

Blocked goroutines don't use a thread.

Channels

Channels provide communication between and within goroutines.

```
c := make(chan string)

// goroutine 1
c <- "hello!"

// goroutine 2
s := <-c
fmt.Println(s) // "hello!"
```

Select

A `select` statement blocks until communication can proceed.

```
select {
case n := <-in:
  fmt.Println("received", n)
case out <- v:
  fmt.Println("sent", v)
}
```

Example

```go
package main

import ("fmt"; "time"; "math/rand")

func main() {
    table := make(chan string)
    for _, player := range [...]string{"alice", "bob"} {
        go func(who string) {
            num := 0;
            for {
                ball := <- table;
                fmt.Printf("player %s: %s\n", who, ball)
                table <- fmt.Sprintf("tick-%s-%d",who,num)
                time.Sleep(time.Millisecond * time.Duration(rand.Intn(1000)))
                num++
            }
        }(player)
    }
    table <- "go"
    time.Sleep(10 * time.Second)
}
```

Run

Example

```
package main

import ("fmt"; "time"; "math/rand")

func main() {
    table := make(chan string)
    for _, player := range [...]string{"alice", "bob", "chris"} {
        go func(who string) {
            num := 0;
            for {
                ball := <- table;
                fmt.Printf("player %s: %s\n", who, ball)
                table <- fmt.Sprintf("tick-%s-%d",who,num)
                time.Sleep(time.Millisecond * time.Duration(rand.Intn(1000)))
                num++
            }
        }(player)
    }
    table <- "go"
    time.Sleep(10 * time.Second)
}
```

Run

Out of the box tools

- go tool
  - Fetching packages
  - Building, installing
  - Instrumenting (race detection
  - Running tests (also with coverage)
  - Static code analysis
  - …
- gofmt and goimports
- Godoc
- cgo for linking with legacy (a.k.a. C-code)

64-bit x86
32-bit x86
32-bit ARM

darwin
dragonfly
freebsd
linux
netbsd
openbsd
plan9
solaris
windows

Runtime

- Go is compiled/linked into machine code (also cross-compiling)
- Executuable embeds type information for introspection
- Executable does not use dynamic/shared libraries (i.e. all static)

Performance

Fastest

Pure C

Almost as fast

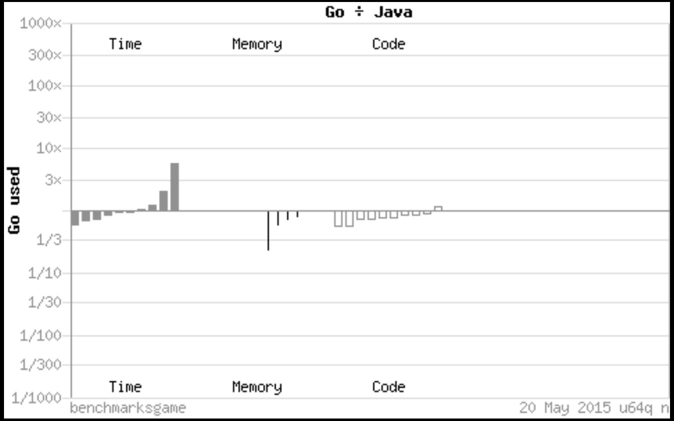
- C++
- Rust
- Go
- Java/Scala

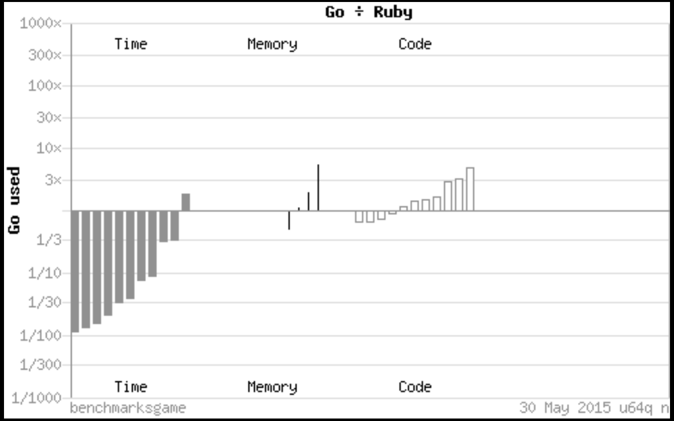
Slower

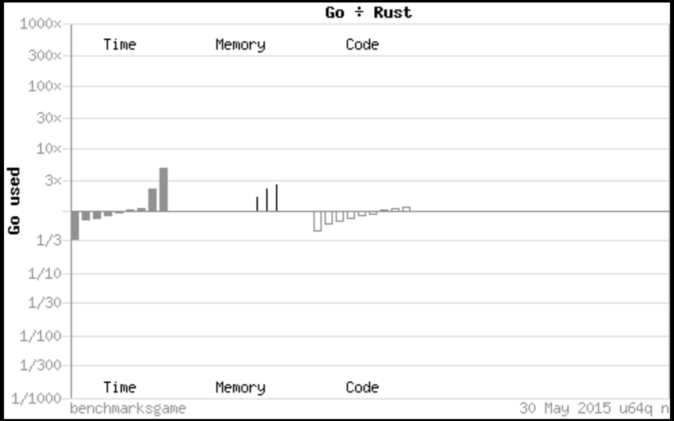
- Node.js
- Ruby
- Python
- Tcl
- PHP

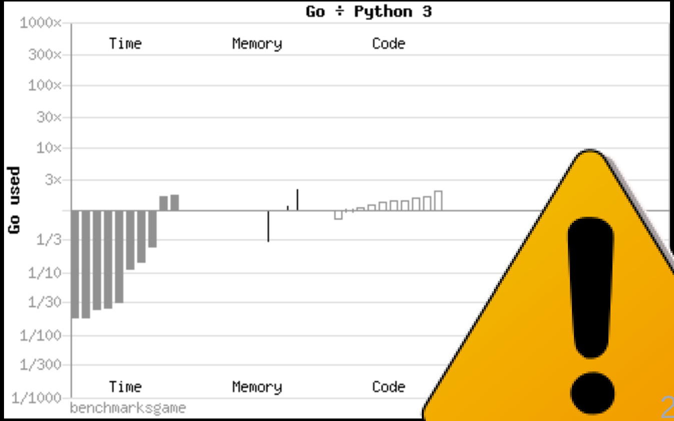
Go executes with similar speed as Java, but uses usually considerably less memory

http://benchmarksgame.alioth.debian.org/

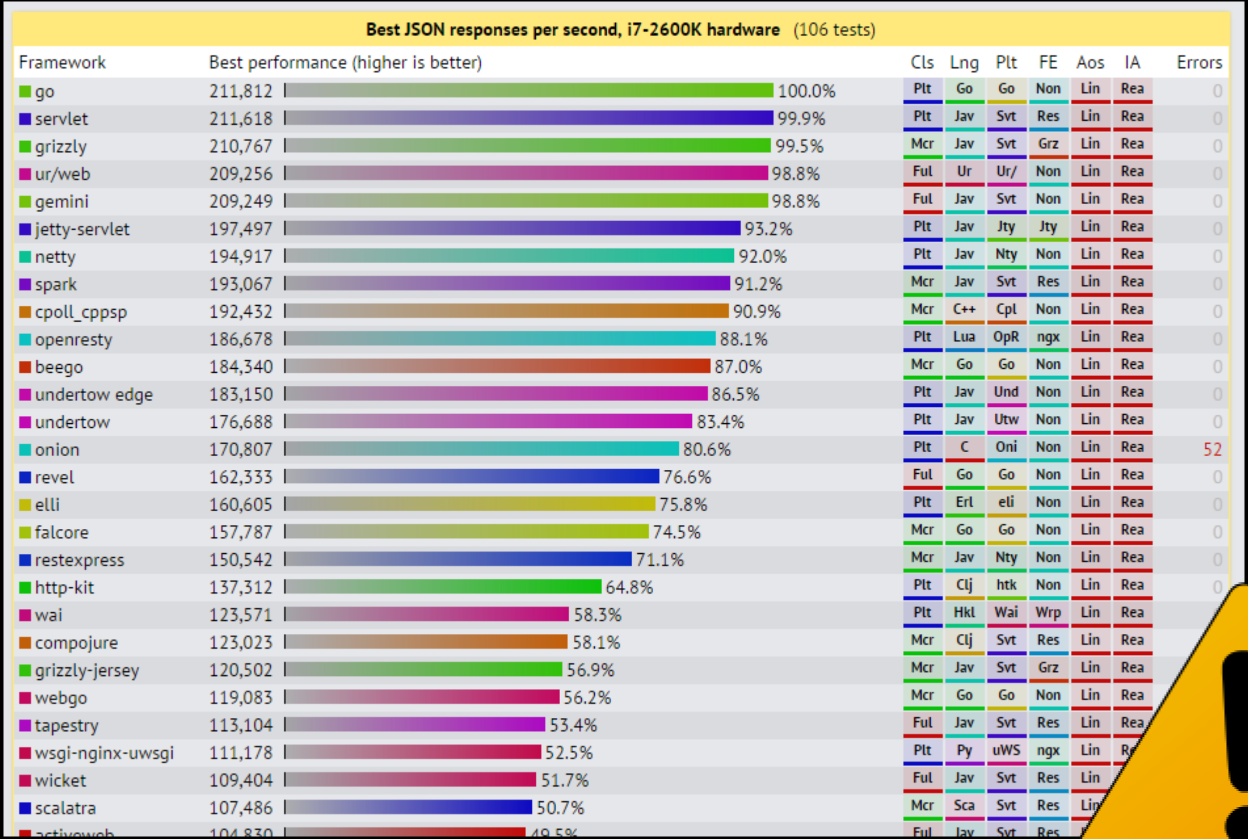








http://www.techempower.com/benchmarks



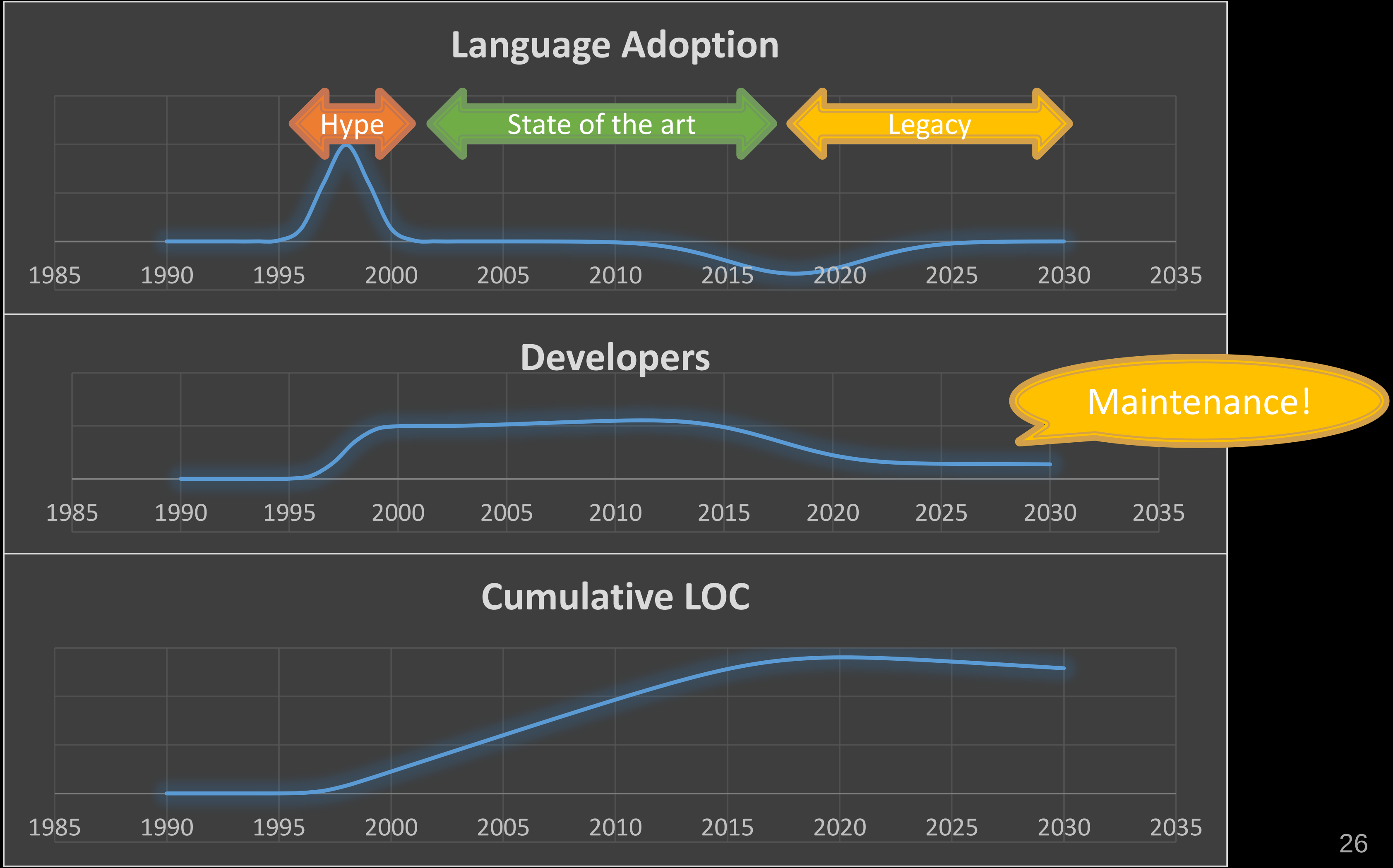
Best JSON responses per second, i7-2600K hardware (106 tests)

Framework	Best performance (higher is better)		Cls	Lng	Plt	FE	Aos	IA	Errors
go	211,812	100.0%	Plt	Go	Go	Non	Lin	Rea	0
servlet	211,618	99.9%	Plt	Jav	Svt	Res	Lin	Rea	0
grizzly	210,767	99.5%	Mcr	Jav	Svt	Grz	Lin	Rea	0
ur/web	209,256	98.8%	Ful	Ur	Ur/	Non	Lin	Rea	0
gemini	209,249	98.8%	Ful	Jav	Svt	Non	Lin	Rea	0
jetty-servlet	197,497	93.2%	Plt	Jav	Jty	Jty	Lin	Rea	0
netty	194,917	92.0%	Plt	Jav	Nty	Non	Lin	Rea	0
spark	193,067	91.2%	Mcr	Jav	Svt	Res	Lin	Rea	0
cpoll_cppsp	192,432	90.9%	Mcr	C++	Cpl	Non	Lin	Rea	0
openresty	186,678	88.1%	Plt	Lua	OpR	ngx	Lin	Rea	0
beego	184,340	87.0%	Mcr	Go	Go	Non	Lin	Rea	0
undertow edge	183,150	86.5%	Plt	Jav	Und	Non	Lin	Rea	0
undertow	176,688	83.4%	Plt	Jav	Utw	Non	Lin	Rea	0
onion	170,807	80.6%	Plt	C	Oni	Non	Lin	Rea	52
revel	162,333	76.6%	Ful	Go	Go	Non	Lin	Rea	0
elli	160,605	75.8%	Plt	Erl	eli	Non	Lin	Rea	0
falcore	157,787	74.5%	Mcr	Go	Go	Non	Lin	Rea	0
restexpress	150,542	71.1%	Mcr	Jav	Nty	Non	Lin	Rea	0
http-kit	137,312	64.8%	Plt	Clj	htk	Non	Lin	Rea	0
wai	123,571	58.3%	Plt	Hkl	Wai	Wrp	Lin	Rea	
compojure	123,023	58.1%	Mcr	Clj	Svt	Res	Lin	Rea	
grizzly-jersey	120,502	56.9%	Mcr	Jav	Svt	Grz	Lin	Rea	
webgo	119,083	56.2%	Mcr	Go	Go	Non	Lin	Rea	
tapestry	113,104	53.4%	Ful	Jav	Svt	Res	Lin	Rea	
wsgi-nginx-uwsgi	111,178	52.5%	Plt	Py	uWS	ngx	Lin	Re	
wicket	109,404	51.7%	Ful	Jav	Svt	Res	Lin		
scalatra	107,486	50.7%	Mcr	Sca	Svt	Res	Lin		
activeweb	104,830	49.5%	Ful	Jav	Svt	Res			

Ecosystem

- Many web frameworks
- Many proposals for language extensions (about 50 proposals for generics ;-))
- Not yet established at Universities for teaching
- No formal certifications yet
- Active community
- Many commercial users, mostly for «heavy-duty» Web work

SO WHAT?


Language Adoption
Hype
State of the art
Legacy
1985
1990
1995
2000
2005
2010
2015
2020
2025
2030
2035
Developers
Maintenance!
1985
1990
1995
2000
2005
2010
2015
2020
2025
2030
2035
Cumulative LOC
1985
1990
1995
2000
2005
2010
2015
2020
2025
2030
2035

RANGERS

In >> Out

Out

From time to time, it is good to clean the desk

Change to where the future will be

Go is sufficiently new, very pragmatic, fast to use, based on experience and not to fancy

If you can, go go.

Resources

Learning

http://tour.golang.org

http://golang.org/wiki/LearnCommunity

Community

http://golang.org/project