GOLANG

George Burgess

Who am I?

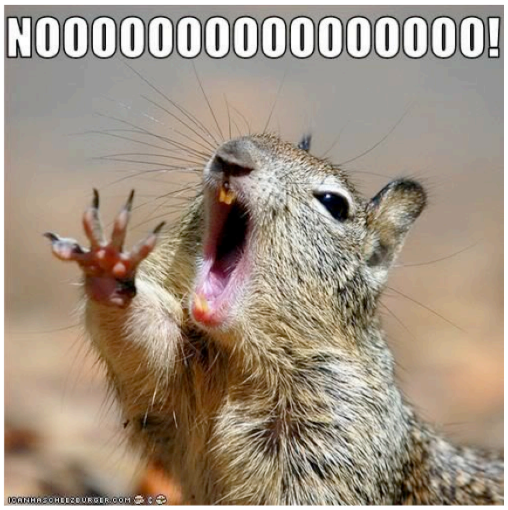
- George Burgess
- Virginia Tech Student
- Likes Go
- Sarcastic
- Please take no offense to anything in this presentation
  - I'll be making fun of languages. ☺
  - I may accidentally bad words

What is Go?

- Programming language
- Created as “modern C” (AHHHH)
- Feels Pythonic
- Made by some of the people who made UNIX
  - And C
- Pioneered by Google

What is Go?

- Systems language



SFKJDSLJFDSFLDJ

SYSTEMS LANGUAGE?!?!
GEORGE.
WE ARE MAKING A WEB CRAWLER.
AS IN. STRINGS. EVERYWHERE.

“Systems Language”

What is Go?

- Garbage Collected
  - Free()dom!
- Tons and tons and tons of built-in libs
  - You'll see in 10 minutes. Or 30.

What *else* is Go?

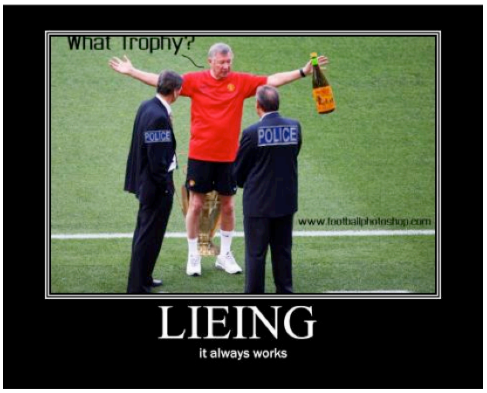
- Actually threaded
  - No GIL!
    - #CPython #MRI
- Fast
  - Fast
    - Did I mention fast?

What *else* is Go?

- Fast
- **Smart**
  - Code snippet!

Ok I lied, no code snippet

- But I will code snippet soon.



Types.

- Go is statically typed.
- Not stupidly though
  - Not stupidly as in Java
  - Not stupidly as in C
  - Not stupidly as in PHP

As in Java?

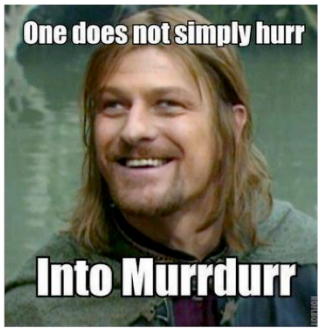
```
Map<String, ArrayList<AbstractAdapterFactoryFacade>> impls = new
Map<String, ArrayList<AbstractAdapterFactoryFacade>>();
```

As in C?

```
int a = -1;
char* ch = (char*)&a;
puts(ch);
```

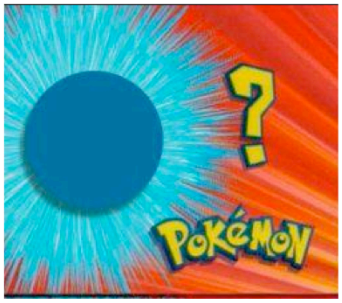
As in PHP?

- `"42abc" == 42 // TRUE`
- `"042" == 042  // FALSE`
- `NULL < -1     // TRUE`
- `NULL == 0     // TRUE`



As in C?

- Let's play a game!



In C…

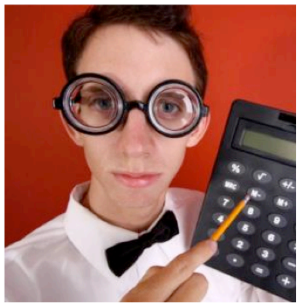
- What's the type of int** foo[];
- Array of pointers to pointers to ints.
  - What.
  - Yes.
  - Array.
  - Of pointers.
  - To pointers.
  - To ints.

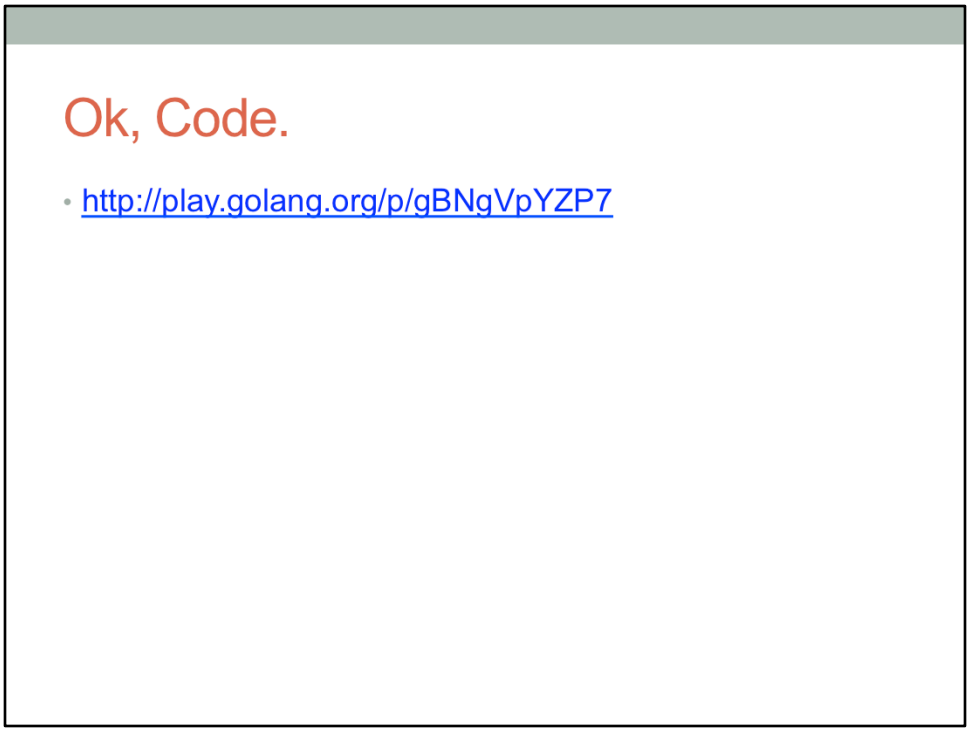
WHY.

- “I mean, if you read the whole thing **right-to-left**, it makes perfect sense!”
  - When the majority of the language is read left-to-right

Go is smarter than this.

- Want to say array of pointers to pointers to integers?
- []**int
  - Array.
  - Of pointers.
  - To pointers.
  - To integers.
  - EZPZ





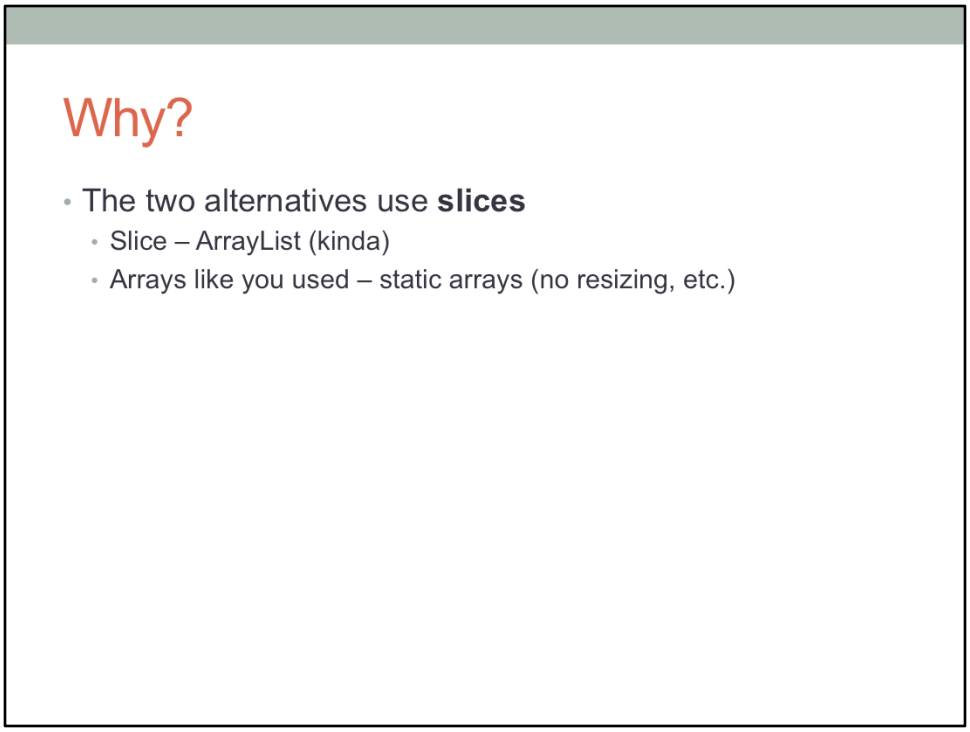
If you want to run the code there locally, copy it to a text file on your local machine. You can then execute `go run filename.go` and Go will compile and run your program for you.

Coding exercise

- This can be done on play.golang.org if you want. Or it can be done local.
- Make an array of size 20
- Initialize with numbers [0..19]
- Run code

Now that you know how not to do it…

- This was probably your code:
  - http://play.golang.org/p/iyhhR-C7b8
- This is how it should be:
  - http://play.golang.org/p/dXKpPTz0QY
  - Or even: http://play.golang.org/p/I6S12y3x1a



Slices are used *everywhere*. They auto-resize on demand. It's great.

Also note that len(slice) and cap(slice) give length and capacity of a slice. Pythonic slice splitting a la:
A := []int{1, 2, 3}
A[1:3]
Is allowed.

Channels

- Strongly typed one-way communication channels
  - Kind of like typed pipes
  - Kind of like typed one-way sockets
  - Kind of like SharedLinkedBlockingQueueFactoryAdaptorInstantiator
    - …Or Queue<T>.

Are we human or are we programmer?

- http://play.golang.org/p/dgYXo48hPy

Why not just use arrays/normal queues?

- Goroutines!

Wat.

- Goroutine – Go-talk for coroutine.
  - Essentially Go threads multiplexed onto actual threads

How do you Goroutine?

- Type `go` in front of a statement. That's it.
- http://play.golang.org/p/znjd-6Qz0W

Why is this cool?

- Well, channels are made so Goroutines can talk
- Also, that code a slide above was running on one thread
- JS users:
  - “Callbacks without the callbacks”

Sooooo…?

- Any of the following will cause a goroutine switch:
  - Blocking Read()
  - Blocking Write()
  - Mutex Lock()
  - Sem Wait()
- …So you can lock 100 mutexes on one thread
  - And it can **still do things**

Yo wan sum concurrency?

- Fibonaccis!
  - Because in FP world, this is the only thing that matters ever
- http://play.golang.org/p/34OmDsIGrg

Benchmarks

- "Lies, damned lies, and benchmarks…"
  - Let's look at one anyway

Ran benchmark.go. You can find it at https://github.com/dyreshark/gotalk/blob/master/benchmark.go

See tl;dr on next slide

Yup.

- Single core of a ULV i5.
  - Goroutine “context switching” took < 300ns
    - Normal thread context switching takes ~20-30us
      - i.e. ratio is 100:1
  - Goroutines are really cheap in terms of memory used
    - They’re also really fast to spawn
- Moral of the story?
  - **Use goroutines. They’re cheap**

Btdubs

- What does the Google say?
  - They run >100,000 Goroutines on one box, no problem
- Some other company reports the same
- We just ran 50,000 of them on an ultrabook.

Some basic tasks…

- Get a webpage
  - Has to be done locally – play.golang.org has net/http disabled ☹
- #CrawlerHype

Oh. Right.

- It might be useful to know how to do stuff with Go
  - `go run`
  - `go get`
  - `go vet`
  - `go fmt`
  - `go test`
  - Debugging?

Go run

- Compiles and runs the target
  - Showed it earlier.

Go get

- Retrieves a repo from a git or mercurial source.
  - Allows you to import it by source name

$ go get github.com/yolo/swag

…Lets me do

```
import "github.com/yolo/swag"

func main() {
  swag.ServeYolo()
}
```

Go vet

- -Wextra for Go
  - Warns of dead code, infinitely recursive string() methods, etc.

Go fmt

- Webcat's worst nightmare
- Formats your code to match One True Style
  - i.e. whatever the Go creators thought was best
  - Also lines things up; see: http://play.golang.org/p/P9FYHgUUKh

Go test

- Unit testing built in.
- Any file *_test.go is considered a test file
- Test methods have signature

```
func TestXxx(*testing.T)
```

- Benchmarks have signature

```
func BenchmarkXxx(*testing.B)
```

- Run `go test [pkgname]` to run tests.

Debugging?

- Gdb. Just like C/C++.
  - There are also graphical debuggers!
    - I have 0 experience with them

Did you set up your GOPATH?

- If not, here's how:

```
mkdir -p ~/go/{src,bin,pkg}
export GOPATH=~/go
```

- If you use BASH, also put that last line in .bashrc
- If you use ZSH, also put that last line in .zshrc
- If you use anything else, gl.

So let's try this out…

- Let's make a goroutine that counts to 5
- Let's make main wait on the goroutine finishing
  - You'll probably need a channel for that.

Pretty solution

- http://play.golang.org/p/4KYrED52wn

Ok let's do stuff that matters

- First, let's grab a webpage and display it to stdout.
- **You can't use play.golang.org to do this**
- It takes like, 8 lines.

Boilerplate

- Get boilerplate at http://play.golang.org
- You'll need net/http (for http.Get)
- You'll need fmt (for fmt.Println)
- You'll want io/ioutil (for ioutil.ReadAll)
- You'll want to use string() on the output from ioutil
- Play around! Ask questions! You have 5-8 minutes.
- You may get any website you want
  - If you don't have one in mind, I suggest http://georgeburgess.info
    - It's a terrible website I had to make for a class. So sue me.

The result looked like: http://play.golang.org/p/Mfd7GAstVE
You CAN NOT run this on play.golang.org. It WILL NOT work.

Cool. Now let's do one better.

- First,

  `go get code.google.com/p/go.net/html`
- Now let's read a few docs.
  - Your goal is to print each "anchor" tag on the site you got.
  - If you want ezpz one, http://georgeburgess.info

Example code

- http://play.golang.org/p/SwzYfDABuB

…And now

- Let's make it modular!
  - i.e. stick it into a function.
  - Function should have signature:
    - func fetchUrl(url string) {
      }

…Finally…

- We have a parallel web crawler. Just put `go` in front of our function. And call it on http href.

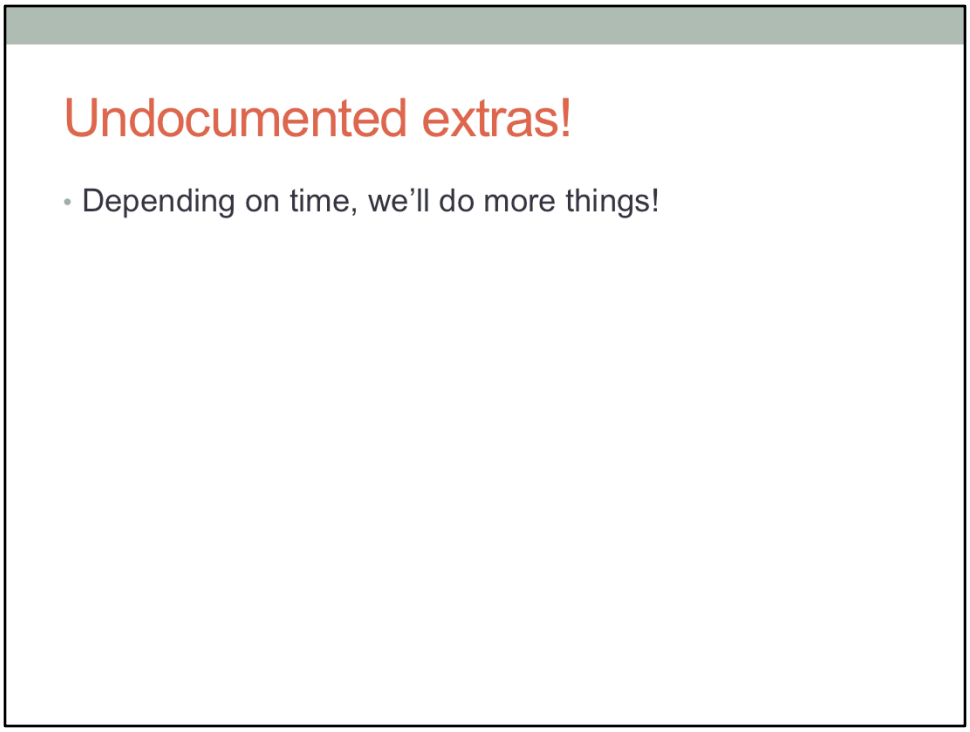
I LIED

- This is concurrent

Let's make it parallel

```
import "runtime"
func main() {
        runtime.GOMAXPROCS(4)
}

// It will now run with 4 processes.
```



Went over the amazingness that is interfaces in Go.

Q/A

- Anyone have questions?
- Anyone?
- Please? ☺

No summary, because we're probably not done!

- …But if you want to leave, you can. I don't mind.

Feel free to say hi!

- george.burgess.iv@gmail.com
- http://github.com/dyreshark
- George Burgess on Facebook
- dyreshark on freenode
  - freenode/#go-nuts is official Go IRC channel