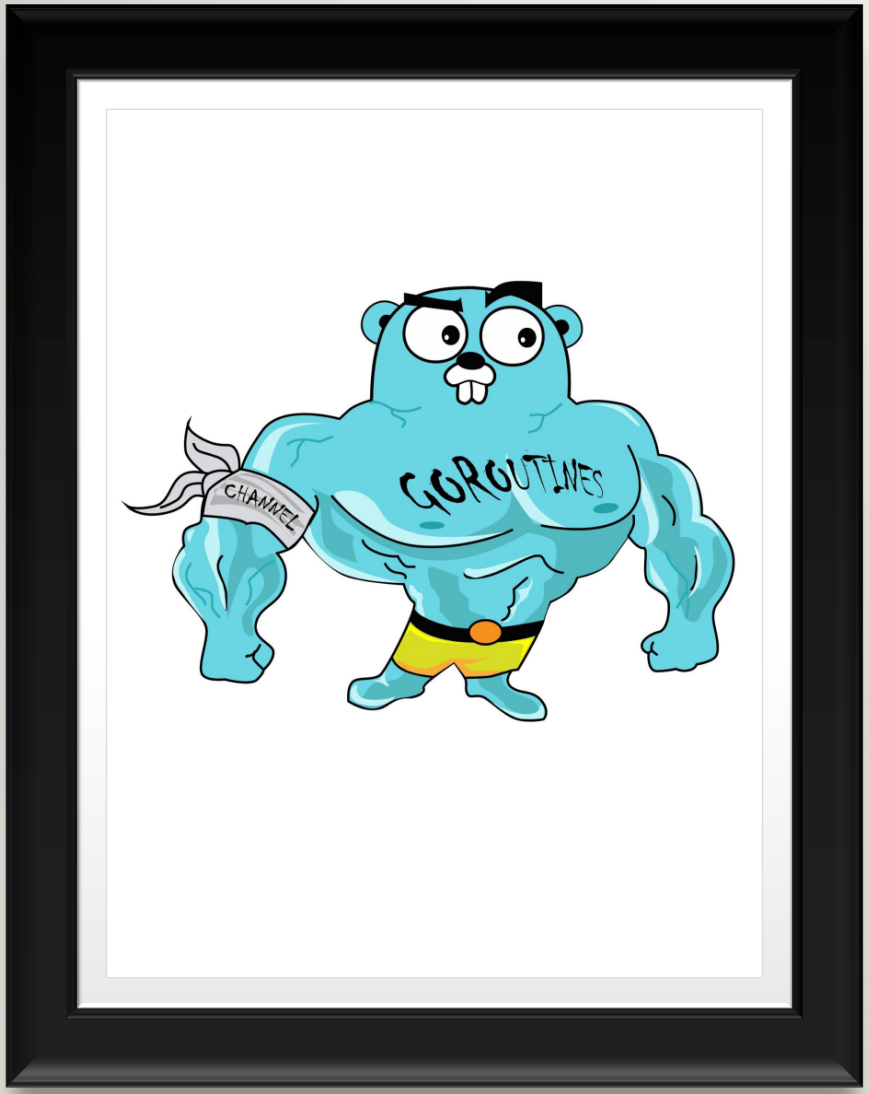


INTRODUCTION TO GOLANG

LIAM HAMPTON

@LiamConroyH

WHATS THE HISTORY?

- Name derived from the animal
- Created by Google Engineers and launched in 2009
- Statically typed language for the multi core processor

@LiamConroyH

FEATURES

- Concurrency logic
- Minimalistic approach = simple
- Type-safe – code only has access to authorized memory locations!
- Garbage collected – frees up memory / less chance of a memory leak
- Fast compilation

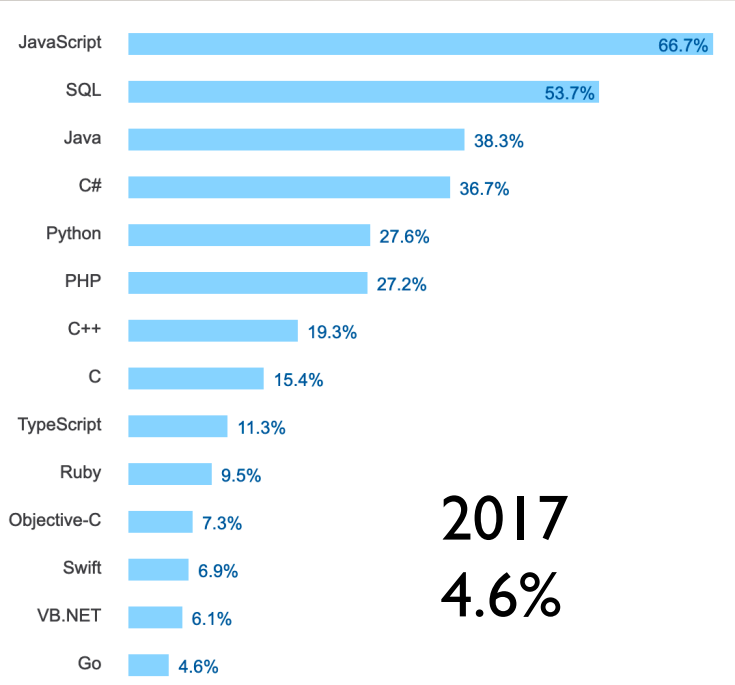
@LiamConroyH

FEATURES

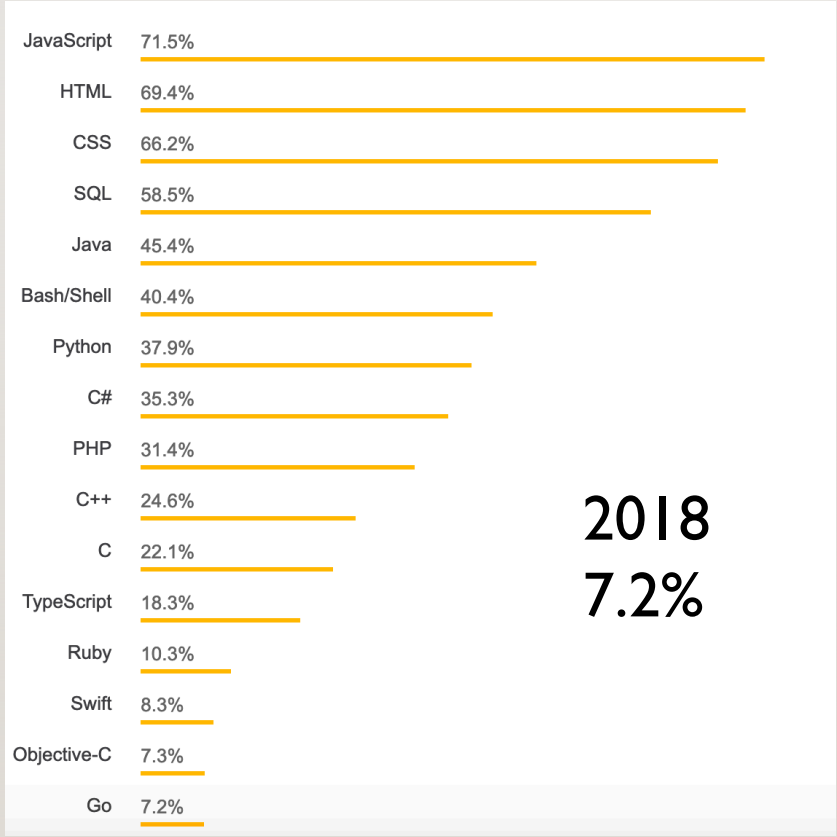
- Cross platform – compiles into binary code
- No classes, it uses Structs instead
- No Inheritance
- Multiple return values (e.g return data, nil)
- Built in tools to test and benchmark

@LiamConroyH

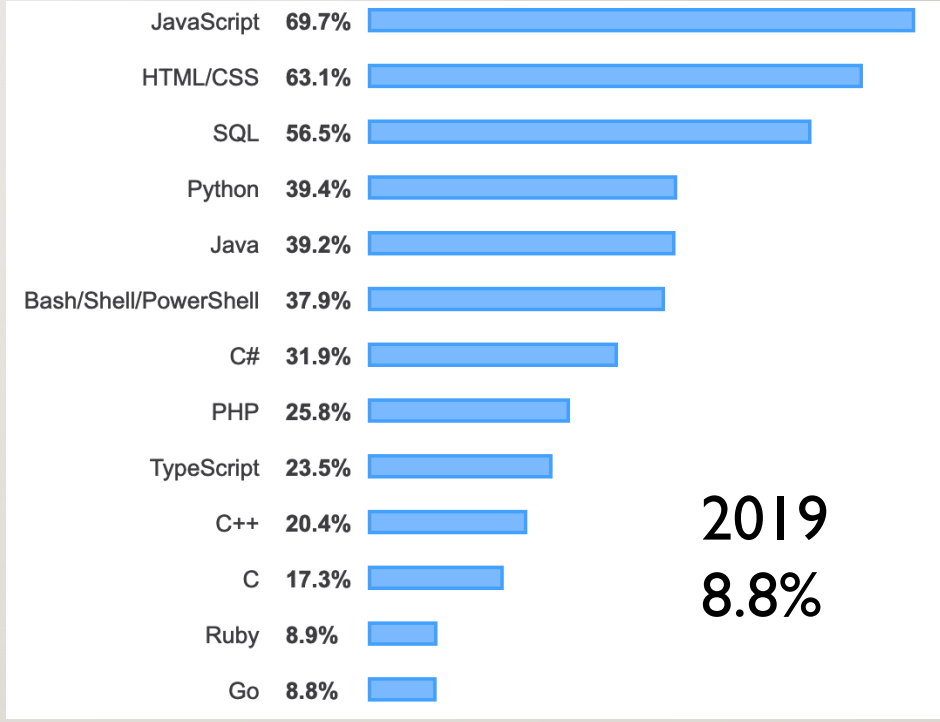
POPULARITY
(AS PER STACK OVERFLOW DEVELOPER SURVEY)



2017
4.6%



2018
7.2%



2019
8.8%

@LiamConroyH

WHERE IS IT BEING USED?

- Google – Lots of projects across the board (Kubernetes, Chrome, Proxy for YouTube etc)
- Uber – Geofence service
- Twitch – Used for the most loaded systems
- Dropbox – Used to scale its systems
- SoundCloud – Used for static analysis in realtime
- Docker

@LiamConroyH

WHY IS IT BEING USED?

- It is a great server-side language for infrastructure
- Easy to scale up with
- Easy package management
- Smart standard library – includes most of what you'll need to get started right out the box

@LiamConroyH

WHAT IS A STRUCT?

- Collection of fields
- Composite data type
- Similar to classes in OO languages
- They can also have functions, just like classes

```
12    type Credentials struct {
13        ConsumerKey       string
14        ConsumerSecret    string
15        AccessToken       string
16        AccessTokenSecret string
17    }
```

```
23    func GetCredentials() Credentials {
24        creds := Credentials{
25            AccessToken:       os.Getenv("ACCESS_TOKEN"),
26            AccessTokenSecret: os.Getenv("ACCESS_TOKEN_SECRET"),
27            ConsumerKey:       os.Getenv("CONSUMER_KEY"),
28            ConsumerSecret:    os.Getenv("CONSUMER_SECRET"),
29        }
30
31        return creds
32    }
```

@LiamConroyH

CONCURRENCY

- Its super easy to create a new thread
- By using the go construct before the function executes it executes it within a new thread - commonly known as a go-routine

@LiamConroyH

CHANNELS

- The magic that connects concurrent go-routines
- Easy to send values into channels from one go-routine and receive those in another go-routine
- To pass an object to a channel we use <- operator on the **right** side
- To read off of the channel we use <- on the **left** side

@LiamConroyH

CROSS-PLATFORM YOU SAY?

- When you build/install an executable is created - code is compiled into binary
  - "go build –o test" creates an executable of your code in the current dir called test
  - "go install –o test" creates an executable of your code in the /bin dir (accessable from anywhere if its on your GOPATH"
- All imports are included
- Because of this, the host system does not need a runtime to execute the binary!

@LiamConroyH

WHAT ARE WE DOING IN THIS WORKSHOP?

Build a simple Golang application

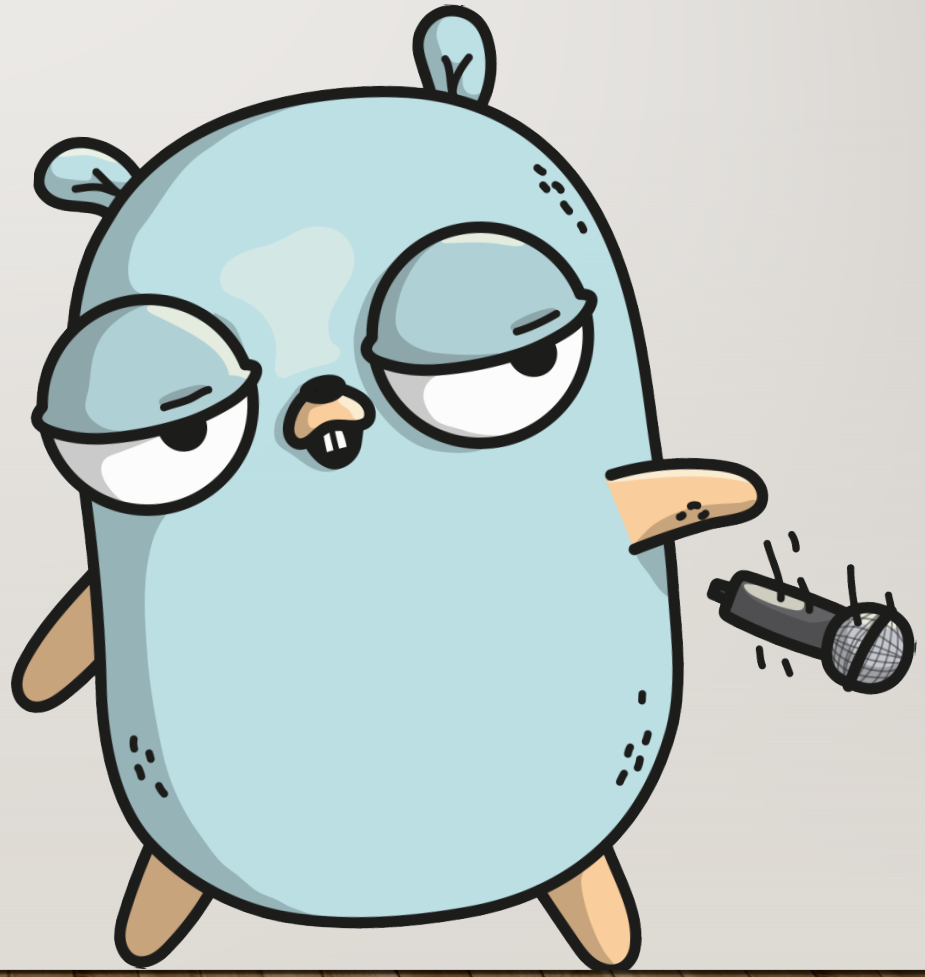
Deploy app into the cloud

Call an API

Call Twitter API & redeploy app

@LiamConroyH

WORKSHOP
LETS CODE



@LiamConroyH