GEOBIN.IO

OVERVIEW AND LESSONS LEARNED

http://github.com/esripdx/geobin-talk/

ESRI PORTLAND R&D CENTER

Courtland Fowler
@FowlerCourt

Josh Yaganeh
@hsoj

Ryan Arana
@aranasaurus

Nate Goldman - UI/UX + general javascript wizardry
@ungoldman

IT'S LIKE REQUESTBIN...
WITH A MAP

MINI DEMO

WHY DID WE MAKE IT?

» "RequestBin is cool, but..."

» Debugging/Visualizing the Geotrigger Service as we built other tools around it

DEMO!

GEOTRIGGER EDITOR > FAKER > GEOBIN

COMPONENTS

» Parsing the Data

» Websockets

» Mocking Redis for tests

» Rate-limiting Middleware

» Deployment Process

PARSING THE DATA

HTTP POST:

```
[{
    "title": "I've got geojson",
    "data": { "message": "Standards rule!" },
    "geo": {
        "type": "Point",
        "coordinates": [-122.6763, 45.5218]
    }
}, {
    "title": "I've got my own geo data schema",
    "data": { "message": "ohai!" },
    "geo": {
        "lat": 45.5218,
        "lng": -122.6763,
        "rad": 100
    }
}]
```

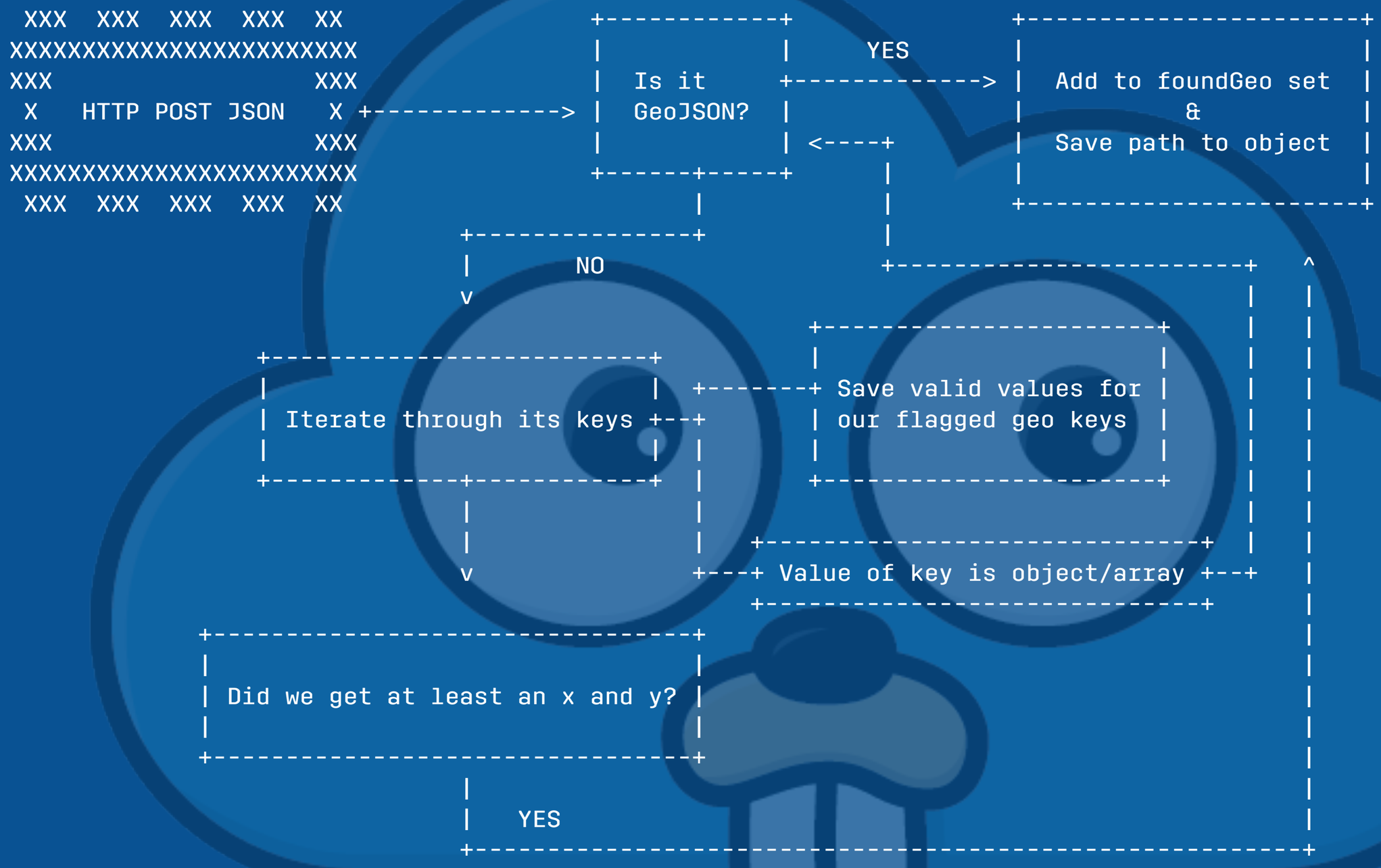
ARBITRARY JSON

HOW WE DETERMINE IF AN OBJECT HAS GEO INFORMATION IN IT

```
{
  "foo": "bar",
  "x": -122.6366, // 'long', 'lng', 'longitude'
  "y": 45.5264,   // 'lat', 'latitude'
  "radius": 100   // 'accuracy', 'acc', 'distance', 'dist', 'rad'
}

// or...

{
  "geo": [-122.6366, 45.5264] // 'coord[s]', 'coordinate[s]', 'loc', 'location'
}
```


XXX XXX XXX XXX XX
XXXXXXXXXXXXXXXXXXXXXXX
XXX XXX
X HTTP POST JSON X
XXX XXX
XXXXXXXXXXXXXXXXXXXXXXX
XXX XXX XXX XXX XX
Is it GeoJSON?
YES
Add to foundGeo set & Save path to object
NO
Iterate through its keys
Save valid values for our flagged geo keys
Value of key is object/array
Did we get at least an x and y?
YES

CHANNELS

» WaitGroup and channel per request

» Fire up a goroutine that just waits for geo data to come in from the channel and adds it to an in-memory slice when it comes in.

» Parse method fires up a goroutine each time it's called and calls itself recursively to find any geo data in the object, sending it through the channel when it does.

WHAT WAS WRONG WITH THAT?

» Race conditions!

» Goroutines sticking around forever because of unclosed channels.

MUTEX

» Each request calls Parse and then waits on the WaitGroup

» Parse method adds 1 to the WaitGroup then fires up a goroutine and parses the object it was given, recursively calling itself until it gets to all of the leaf nodes.

» This time, when it finds a geo object it calls a method which locks the mutex, appends to the found geo slice, unlocks and moves on.

WEBSOCKETS (YAY!)

github.com/gorilla/websocket

SOCKET STRUCT

```
s := &s{
    name:      name,
    ws:        ws,
    send:      make(chan []byte, 256),
    shutdown:  make(chan bool),
    closed:    false,
    closeLock: &sync.Mutex{},
    onRead:    or,
    onClose:   oc,
}

go s.writePump()
go s.readPump()
return s
```

TRIMMING DOWN THE INTERFACE

```
// S exposes some methods for interacting with a websocket
type Socket interface {
    // Submits a payload to the web socket as a text message.
    Write([]byte)
    // return the provided name
    GetName() string
    // Close the socket.
    Close()
}
```

KEEPING TRACK OF SOCKETS

```
type SocketMap interface {
    Add(binName, socketUUID string, s Socket)
    Get(binName, socketUUID string) (Socket, bool)
    Delete(binName, socketUUID string) error
    Send(binName string, payload []byte) error
}
```

CONCURRENCY WITH REDIS!

```
// redis client
client := redis.NewTCPClient(&redis.Options{
    Addr:     conf.RedisHost,
    Password: conf.RedisPass,
    DB:       conf.RedisDB,
})

// redis pubsub connection
ps := client.PubSub()

// prepare a socketmap
sm := NewSocketMap(ps)

// loop for receiving messages from Redis pubsub, and forwarding them on to relevant ws connection
go redisPump(ps, sm)

defer func() {
    ps.Close()
    client.Close()
}()
```

REDIS MOCKING

» Recall Francesc's best practices!

Testing

Using an interface instead of a concrete type makes testing easier.

```
package drawer

import (
    "math"
    "testing"
)

type TestFunc func(float64) float64

func (f TestFunc) Eval(x float64) float64 { return f(x) }

var (
    ident = TestFunc(func(x float64) float64 { return x })
    sin   = TestFunc(math.Sin)
)

func TestDraw_Ident(t *testing.T) {
    m := Draw(ident)
    // Verify obtained image.
```

INTERFACES MAKING TESTING EASIER!

```
// mock our use of redis pubsub for modularity/testing purposes
type PubSubber interface {
    Subscribe(channels ...string) error
    Unsubscribe(channels ...string) error
}

// mock our use of redis client for modularity/testing purposes
type RedisClient interface {
    ZAdd(key string, members ...redis.Z) (int64, error)
    ZCount(key, min, max string) (int64, error)
    Expire(key string, dur time.Duration) (bool, error)
    Publish(channel, message string) (int64, error)
    ZRevRange(key, start, stop string) ([]string, error)
    Exists(key string) (bool, error)
    Get(key string) (string, error)
    Incr(key string) (int64, error)
}
```

TESTS WITHOUT DEPENDENCIES

```
type MockRedis struct {
    sync.Mutex
    bins  map[string][]string
    incrs map[string]string
}

type MockPubSub struct{}

func NewMockRedis() *MockRedis {
    return &MockRedis{
        bins:  make(map[string][]string),
        incrs: make(map[string]string),
    }
}
```

RATE LIMITING

```
key := fmt.Sprintf("rate-limit:%s:%d", r.URL.Path, time.Now().Unix())

if keyExistsInRedis(key) && requestCount(key) > requestsPerSec {
    http.Error(w, "stahp!", http.StatusServiceUnavailable)
    return
}

incrementRequestCount(key)

// continue serving request
```

BASIC MIDDLEWARE

```
func myMiddleware(h http.HandlerFunc) http.HandlerFunc {
    return func(w http.responseWriter, r *http.Request) {

        // do rate limiting

        h.ServeHTTP(w, r)
    }
}
```

RATE LIMIT MIDDLEWARE

```
func rateLimit(h http.HandlerFunc) http.HandlerFunc {
    return func(w http.responseWriter, r *http.Request) {
        key := fmt.Sprintf("rate-limit:%s:%d", r.URL.Path, time.Now().Unix())

        if keyExistsInRedis(key) && requestCount(key) > requestsPerSec {
            http.Error(w, "stahp!", http.StatusServiceUnavailable)
            return
        }

        incrementRequestCount(key)

        h.ServeHTTP(w, r)
    }
}

r := http.NewServeMux()
r.HandleFunc("/api/1/foo", rateLimit(fooHandler))
```

DEPLOYMENT PROCESS

QUESTIONS?

THANKS!

» Ryan Arana
@aranasaurus | github.com/aranasaurus

» Josh Yaganeh
@hsoj | github.com/jyaganeh

» Courtland Fowler
@FowlerCourt | github.com/courtf

http://geobin.io/
http://github.com/esripdx/geobin-talk/