Serverless APIs in Swift

Rob Allen

April 2017 ~ @akrabat

Let's start with Swift

What's Swift?

Swift is a general-purpose programming language built using a modern approach to safety, performance, and software design patterns.

swift.org

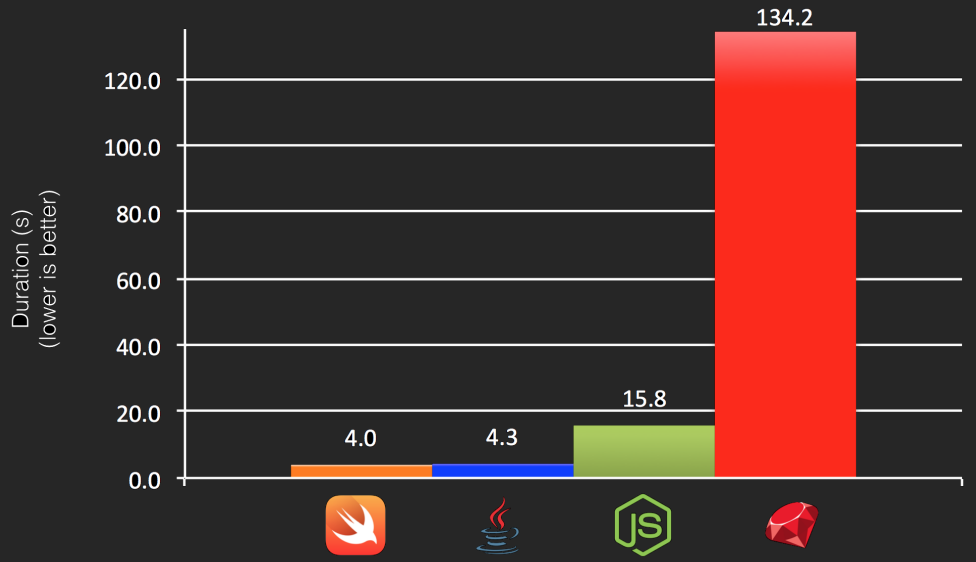
Open Source

- Created by Apple
- Apache 2 license
- Source code on GitHub
- Swift-evolution: open design of new features

Cross Platform

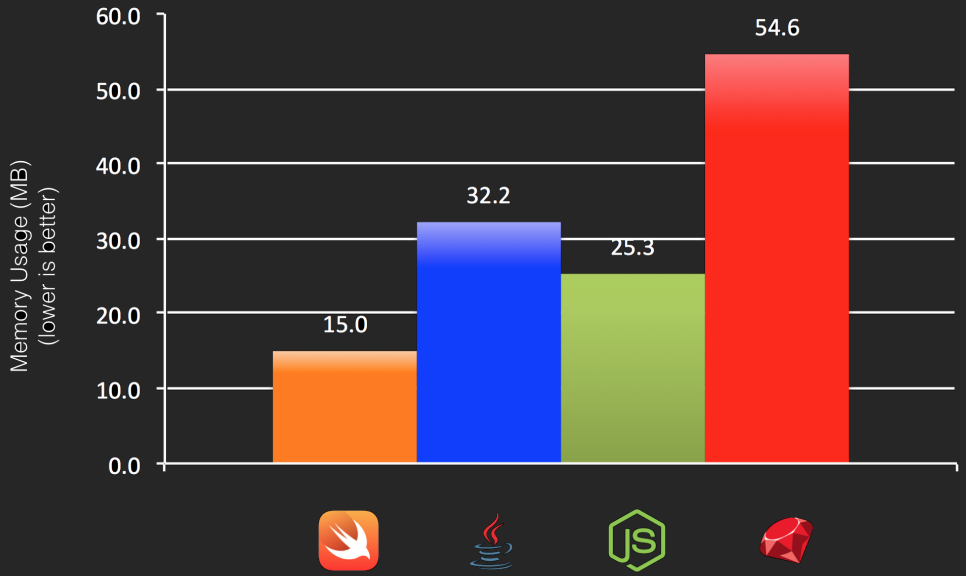
- Runs on Linux (x86) and all Apple OSs
- Ports in progress: Android, Linux(ARM), FreeBSD, Windows
- Libraries: Standard library, Foundation, Dispatch & XCTest

Performance



http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Memory



http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=spectralnorm

Major features

Strong typing
Type inference
Optionals
Closures

Custom operators
Tuples
Generics
Interoperable with C

Safety

- Type safety
- Prefer constants over variables
- Variables are always initialized before use
- Optionals: variables can never be `nil`

Rock-Paper-Scissors

```
 1 import Foundation
 2
 3 let shapes = ["rock", "paper", "scissors"]
 4
 5 for count in 1...3 {
 6     print(count)
 7     sleep(1)
 8 }
 9
10 srandom(UInt32(NSDate().timeIntervalSince1970))
11 let chosenShape = random() % shapes.count
12 print(player[chosenShape]);
```

Result

```
$ swift rock-paper-scissors.swift
1
2
3
scissors
```

Structs

Swift's value objects

```
 1 struct Money {
 2     enum Currency { case GBP, EUR, USD }
 3     let money: (Decimal, Currency)
 4
 5     init (amount: Decimal, currency: Currency) {
 6       money = (amount, currency)
 7     }
 8
 9     var amount: String {
10       get { return money.0.round(to: 2) }
11     }
12 }
```

Structs

Usage:

```
1 let fivePounds = Money(amount: 5.20, currency: .GBP)
2 print(fivePounds.amount)
```

Compile and run:

```
$ swift test.swift
5.20
```

Classes

Swift's reference objects (& you can inherit!)

```
1 class Child {
2     var name: String
3     var age: Int
4
5     init (name: String, age: Int) {
6         self.name = name
7         self.age = age
8     }
9 }
```

Reference vs value types

Classes are reference types:

```
1 var judith = Child(name: "Judith", age: 12)
2 var karen = judith
3 karen.name = "Karen"
4
5 print(judith.name)
6 print(karen.name)
```

```
$ swift test.swift
Karen
Karen
```

Reference vs value types

Structs are value types

```
1 var fivePounds = Money(money: (5.20, .GBP))
2 var tenPounds = fivePounds
3 tenPounds.money = (10.00, .GBP)
4
5 print(fivePounds.amount)
6 print(tenPounds.amount)
```

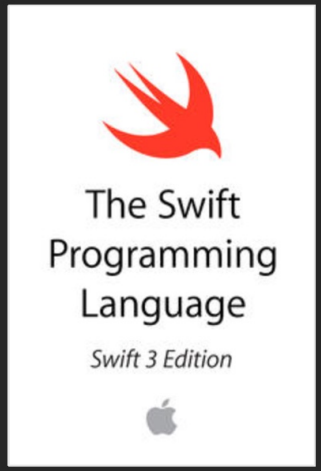
```
$ swift test.swift
5.20
10.00
```

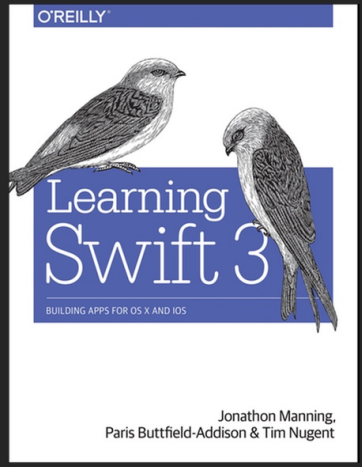
Protocols

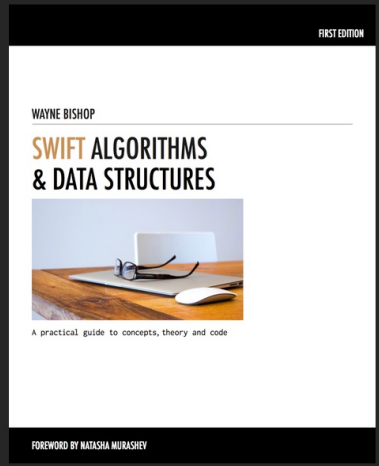
- Blueprint of methods & properties, etc that suit a task
- Protocols are *adopted* by classes & structures

```
1  protocol Shareable {
2    func toJSON() -> String
3  }
4
5
6  extension Money : Shareable {
7    func toJSON() -> String {
8      // implement here
9      return json
10   }
11 }
```

Learn the language







Serverless

Serverless?

The first thing to know about serverless computing is that "serverless" is a pretty bad name to call it.

Brandon Butler, Network World

Serverless

AKA: *Functions as a Service*

- A runtime to execute your functions
- No capacity planning or load balancing; just tasks being executed.
- Pay for execution, not when idle

Use-cases

Synchronous	Service is invoked and provides immediate response (HTTP request)
Asynchronous	Push a message which drives an action later (web hooks, timed events)
Streaming	Continuous data flow to be processed

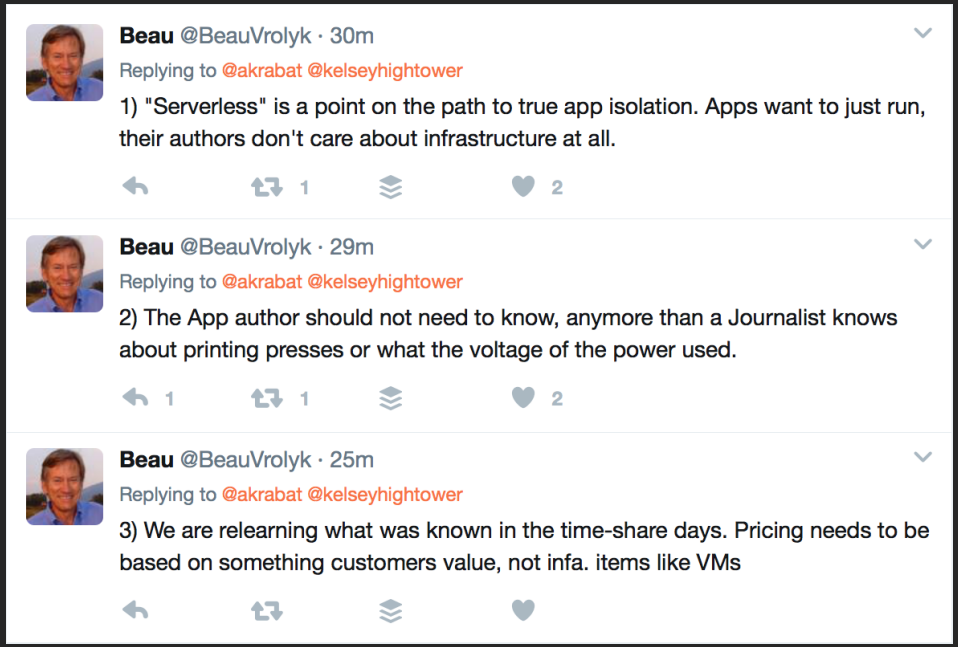
Benefits

- No need to think about servers
- Concentrate on application code
- Pay only for what you use, when you use it
- Language agnostic: NodeJS, Python, Java, Swift, C#, etc

Challenges

- Start up latency
- Time limit
- State is external
- DevOps is still a thing

It's about value



Hello world

(coding time!)

Let's talk about HTTP APIs

HTTP APIs

Just because it's *serverless* doesn't mean we can ignore the basics!

- HTTP method negotiation
- Content-type handling
- Good error handling
- Media type format

What is Rest?

- An *architecture*
- Centres on the transfer of *representations* of *resources*
    - A *resource* is any concept that can be addressed
    - A *representation* is typically a document that captures the current or intended state of a resource
- A *client* makes requests of a server when it wants to transition to a new state

Strengths

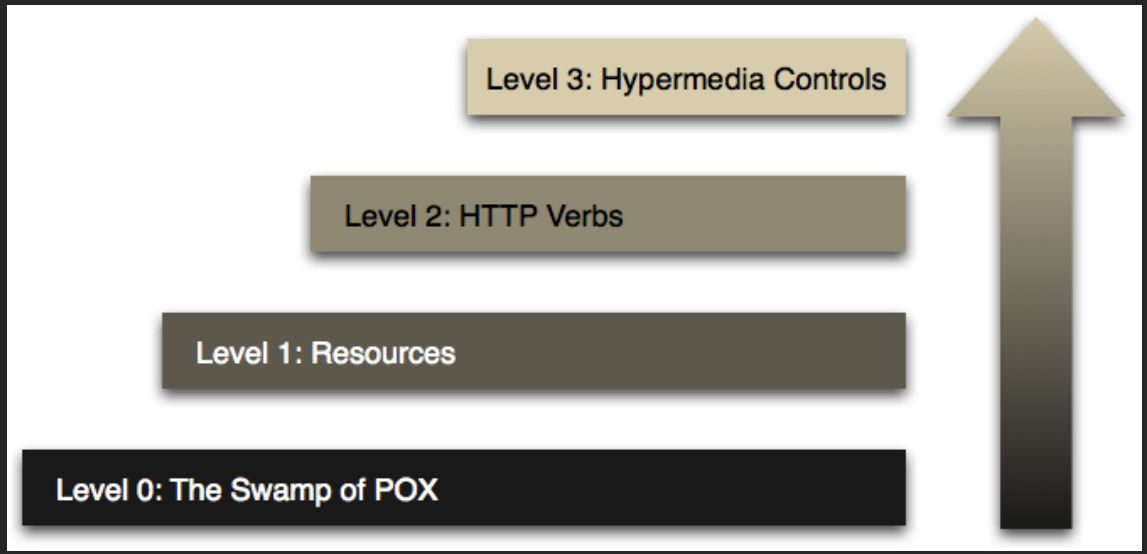
- Loose coupling
- Leverages the power of HTTP
- Emphasis on readability
  - HTTP methods as verbs: GET, POST, PUT, DELETE, etc.
  - Resources as nouns: collections and entities

Constraints

- Client/Server
- Stateless
- Cacheable
- Layered system
- Uniform Interface

    Hypermedia as the engine of application state (HATEOAS)

Richardson Maturity Model



source: http://martinfowler.com/articles/richardsonMaturityModel.html

Primary aspects of a RESTful API

- URI for *each* resource: `https://api.example.com/users/rob`
- HTTP methods are the set of operations allowed for the resource
- Media types are used for representations of the resource
- The API is be hypertext driven

URI for each resource

- Separate endpoint for each resource
- A resource can be a collection e.g. */users*
- or a single entity e.g. */users/rob*

HTTP method negotiation

```
$ curl -i -X PUT http://example.com/ping
HTTP/1.1 405 Method Not Allowed
Allow: GET
Connection: close
Content-Length: 53
Content-type: application/json

{
    "message": "Method not allowed. Must be one of: GET"
}
```

Status codes

Send the right one for the right situation!

1xx	Informational
2xx	Success
3xx	Redirection
4xx	Client error
5xx	Server error

HTTP verbs

Method	Used for	Idempotent?
GET	Retrieve data	Yes
PUT	Change data	Yes
DELETE	Delete data	Yes
POST	Change data	No
PATCH	Update data	No

Content negotiation

Correctly parse the request

- Read the `Content-Type` header
- Raise "415 Unsupported media type" status if unsupported

Correctly create the response

- Read the `Accept` header
- Set the `Content-Type` header

Hypermedia controls

a.k.a: Links between resources.

- Media type used for a representation
- The link relations between representations and/or states
- Important for discoverability
- Options: HAL, Collection+JSON, JSON-LD

application/hal+json

https://tools.ietf.org/html/draft-kelly-json-hal

```
{
  "_links": {
    "self": { "href": "https://example.com/orders/523" },
    "warehouse": { "href": "https://example.com/warehouse/56" },
    "invoice": { "href": "https://example.com/invoices/873" }
  },
  "currency": "GBP",
  "status": "shipped",
  "total": 123.45
}
```

Let's look at APIs

(coding time!)

Summary

Resources

This talk:

- https://github.com/SwiftOnTheServer/flashcards
- https://akrabat.com/talks/#sais

Around the web:

- https://swift.org
- https://openwhisk.org
- https://medium.com/openwhisk

Questions?

Rob Allen ~ @akrabat

Thank you!

Rob Allen ~ @akrabat