Hi, I'm Donny.

I build iOS apps.

Donny Wals
Mastering
iOS 10 Programming
Unlock the full potential of iOS 10 programming with Swift 3.0 to build rich and powerful applications
Packt

Donny Wals
Mastering
iOS 11 Programming
Second Edition
Unlock the potential of iOS programming with Swift to build rich and powerful applications
Packt

JSON and Swift…

Still A Better Love Story Than Twilight

Topics

- Pre-Swift 4.0 JSON handling
- Some things I learned about JSON pre-Swift 4.0
- Handling JSON in Swift 4.0
- Concluding remarks

Pre-Swift 4.0 JSON Handling

JSON File:

```
{
  "items": [{
    "line_up": [{
      "artist": {
        "name": "Foo"
      }
    }]
  }]
}
```

Swift 1.0:

```
if let items = json["items"] as? [[String: AnyObject]] {
    if let firstItem = items.first {
        if let item = firstItem as? [String: AnyObject] {
            if let lineUp = item["line_up"] as? [[String: AnyObject]] {
                if let firstLineupItem = lineUp.first {
                    if let artist = firstLineupItem["artist"] as? [String: AnyObject] {
                        if let name = artist["name"] as? String {
                            print(name)
                        }
                    }
                }
            }
        }
    }
}
```

Pre-Swift 4.0 JSON Handling

SwiftyJSON

```
guard let item = json["items"].arrayValue.first,
    let lineUpItem = item["line_up"].arrayValue.first,
    let name = lineUpItem["artist"]["name"].string
    else { return }

print(name)
```

SwiftyJSON is amazing!

```
guard let item = json["items"].arrayValue.first,
    let lineUpItem = item["line_up"].arrayValue.first,
    let name = lineUpItem["artist"]["name"].string
    else { return }

print(name)
```

A performance test

```
{ id: 260,
  name: 'DAS Foute Oktoberfest 20171',
  start: '2017-02-20T16:00:00+0000',
  end: '2017-10-07T23:59:00+0000',
  publish_from: '2016-11-09T08:00:00+0000',
  publish_to: '2017-10-07T23:59:00+0000',
  categories: [ { id: 2, name: 'NAME GOES HERE' } ],
  location:
   { name: 'Goffertpark, Nijmegen (NL)',
     address: '',
     zipcode: '1234AB',
     city: 'Nijmegen',
     state: 'Noord-whateverland',
     country: 'NL',
     latitude: 51.82548,
     longitude: 5.836629,
     url: 'https://www.google.nl/maps/place/Goffertpark/@52.2576689,5.2644167,8.25z/data=!4m5!3m4!
1s0x47c7089b2ce7dff1:0xfd37f35c5b91b9c8!8m2!3d51.8255586!4d5.8366532',
     email: 'random@email.com',
     id: 169,
     radius: 500 },
  line_up:
   [ { id: 1, name: 'nino' },
     { id: 2, name: 'lisa' },
     { id: 7, name: 'kees' },
     { id: 4, name: 'Rodney' },
     { id: 8, name: 'Oscar' },
     { id: 9, name: 'Dick' } ],
  description: 'Matrixx presenteert Het Foute Oktoberfest 2017!\r\n\r\nZaterdag 7 oktober in een grote feesttent op de Goffertweide in
Nijmegen.\r\n\r\nBinnenkort meer info!',
  image_url: 'http://app.appimin.com/filelib/storage/events/58246fa620305_het-foute-oktoberfest-2017.jpg' }
```

- 2080 events
- Nested objects
- Tested on iPhone 5c

A performance test

```
func loadSwifty() {
    guard let data = getFileData()
        else { return }

    let jsonFile = JSON(data: data)
    for eventJSON in jsonFile["events"].arrayValue {
        let location = Location(id: eventJSON["location"]["id"].intValue,
                                name: eventJSON["location"]["name"].stringValue,
                                lat: eventJSON["location"]["latitude"].double,
                                lon: eventJSON[“location"]["longitude"].double,
                                address: eventJSON["location"]["address"].stringValue,
                                zipcode: eventJSON[“location"]["zipcode"].stringValue)

        // etc.
    }

    showCompletion()
}
```

👎😭

6.76 seconds to complete

A performance test

```swift
func loadNormal() {
    guard let data = getFileData(),
        let jsonObject = try? JSONSerialization.jsonObject(with: data, options: []),
        let json = jsonObject as? DWJSON,
        let events = json["events"] as? [DWJSON]
        else { return }

    for eventJSON in events {
        guard let locationJSON = eventJSON["location"] as? DWJSON,
            let categoriesJSON = eventJSON["categories"] as? [DWJSON],
            let lineUpJSON = eventJSON["line_up"] as? [DWJSON]
            else { return }

        // etc.
    }

    showCompletion()
}
```

1.84 seconds to complete

What did I learn?

- Be careful about libraries that make things easy; they might be (very) slow.
- Sometimes uglier code is faster (unfortunately).
- Working with JSON isn’t as bad as it seems in Swift 2.0+.

What about Swift 4.0?

❓ 🤷‍♂️ ❓

Introducing Codable

< Swift 4.0

```
struct Category {
    let id: Int
    let name: String
}
```

```
let category = Category(id: categoryJSON["id"] as? Int ?? 0,
                        name: categoryJSON["name"] as? String ?? "")
```

Swift 4.0

```
struct Category: Codable {
    let id: Int
    let name: String
}
```

```
let decoder = JSONDecoder()
let category = try? decoder.decode(Category.self, from: data)
```

Property names are directly mapped to JSON keys

Introducing Codable

But what if the keys don't match?

```
location:
  { name: 'Goffertpark, Nijmegen (NL)',
    address: '',
    zipcode: '1234AB',
    city: 'Nijmegen',
    state: 'Noord-whateverland',
    country: 'NL',
    latitude: 51.82548,
    longitude: 5.836629,
    url: 'https://www.google.nl/',
    email: 'random@email.com',
    id: 169,
    radius: 500 }
```

```
struct Location: Codable {
    enum CodingKeys: String, CodingKey {
        case id, name, address, zipcode
        case lat = "latitude"
        case lon = "longitude"
    }

    let id: Int
    let name: String
    let lat: Double?
    let lon: Double?
    let address: String
    let zipcode: String
}
```

```
struct Location {
    let id: Int
    let name: String
    let lat: Double?
    let lon: Double?
    let address: String
    let zipcode: String
}
```

Codable Performance

```
func loadCodable() {
    guard let data = getFileData()
        else { return }

    do {
        let decoder = JSONDecoder()
        let eventsResponse = try decoder.decode(EventsResponse.self, from: data)

        showCompletion()
    } catch {
        print(error)
    }
}
```

👍😍

1.82 seconds to complete

Codable and Date

```swift
struct DWDate: Codable {
    let date: Date?
}

let jsonString = "{\"date\": \"31-08-2017 +0000\"}"
let json = jsonString.data(using: .utf8)!

do {
    let decoder = JSONDecoder()
    let formatter = DateFormatter()
    formatter.dateFormat = "dd-MM-yyyy Z"
    decoder.dateDecodingStrategy = .formatted(formatter)
    let response = try decoder.decode(DWDate.self, from: json)
} catch {
    print(error)
}
```

Codable and Date

```
struct DWDate: Codable {
    let date: Date?
}

let jsonString = "{\"date\": \"\"}" //👈🏼👈🏼👈🏼
let json = jsonString.data(using: .utf8)!

do {
    let decoder = JSONDecoder()
    let formatter = DateFormatter()
    formatter.dateFormat = "dd-MM-yyyy Z"
    decoder.dateDecodingStrategy = .formatted(formatter)
    let response = try decoder.decode(DWDate.self, from: json)
} catch {
    print(error)
}
```

Codable and Date

```
struct DWDate: Codable {
    let date: Date?

    init(from decoder: Decoder) throws {
        let container = try decoder.container(keyedBy: CodingKeys.self)

        self.date = try? container.decode(Date.self, forKey: .date)
    }
}
```

Codable and Date

```
dataCorrupted(Swift.DecodingError.Context(codingPath:
[__lldb_expr_156.DWDate.(CodingKeys in
_995AD5D014A8F9E1965F4BEEB81F4E38).date], debugDescription: "Date
string does not match format expected by formatter.", underlyingError: nil))
```

What else should you know?

```json
{
  "name" : "Test",
  "type" : 1
}
```

```swift
struct Prize: Codable {
    enum PrizeType: Int, Codable {
        case ticket = 1, voucher = 2
    }

    let name: String
    let type: PrizeType
}
```

```swift
let prize = Prize(name: "Test", type: .ticket)
let encoder = JSONEncoder()
let result = try! encoder.encode(prize)
```

```json
{
  "name" : "Test",
  "type" : 1
}
```

What else should you know?

```
struct EventsResponse: Codable {
    let events: [Event]
}
```

```
let eventsResponse = try decoder.decode([String: [Event]].self, from: data)

let eventsResponse = try decoder.decode(EventsResponse.self, from: data)
```

Concluding remarks

- Handling JSON with libraries can be convenient yet slow
- The Codable protocol performs really well
- Optional date handling is a bit too strict IMO
- You can do really powerful things with Codable

http://benscheirman.com/2017/06/ultimate-guide-to-json-parsing-with-swift-4/

Thanks!