SWIFT PARIS JUNIOR

DECEMBER 2019

INTRODUCTION TO CODABLE

FRANK LEFEBVRE

PROGRAM

- Handling JSON
- Other Encoders/Decoders
- Advanced Usages

HANDLING JSON

ONCE UPON A TIME, A NETWORK REQUEST...

```
let session = URLSession(configuration: .default)
let url = URL("https://example.com/movies/14")
let datatask = session.dataTask(with: url) {
    data, response, error in
    if error == nil, let data = data {
        do {
            let movie = try ... // TODO: decode JSON here
            completion(movie)
        }
        catch { ... }
    }
}
datatask.resume()
```

ONCE UPON A TIME, A NETWORK REQUEST...

```
let session = URLSession(configuration: .default)
let url = URL("https://example.com/movies/14")
let datatask = session.dataTask(with: url) {
    data, response, error in
    if error == nil, let data = data {
        do {
            let movie = try ... // TODO: decode JSON here
            completion(movie)
        }
        catch { ... }
    }
}
datatask.resume()
```

WITHOUT CODABLE

- NSJSONSerialization
  - [String: Any]
- Dependencies
  - SwiftyJSON
  - Alamofire
  - ...

CODABLE

- Swift 4.0 (Standard Library)
- Two protocols
  - Encodable & Decodable
- Encoders & Decoders
  - JSONEncoder, JSONDecoder
  - PropertyListEncoder, PropertyListDecoder

DECODING: BASIC USAGE

```
GET "https://example.com/movies/14"
{"id": 14, "title": "Star Wars IV: A New Hope", "director":
"George Lucas", "trailer": "https://example.com/starwars.m3u8"}
```

DECODING: BASIC USAGE

```
GET "https://example.com/movies/14"
{"id": 14, "title": "Star Wars IV: A New Hope", "director":
"George Lucas", "trailer": "https://example.com/starwars.m3u8"}
```

```
struct Movie {
    let id: Int
    let title: String
    let director: String
    let trailer: URL
}
```

DECODING: BASIC USAGE

```
GET "https://example.com/movies/14"
{"id": 14, "title": "Star Wars IV: A New Hope", "director":
"George Lucas", "trailer": "https://example.com/starwars.m3u8"}
```

```
struct Movie: Codable {
    let id: Int
    let title: String
    let director: String
    let trailer: URL
}

let decoder = JSONDecoder()
let movie = try decoder.decode(Movie.self, from: data)
```

CODING STRATEGIES

- JSONDecoder
  - dateDecodingStrategy
  - dataDecodingStrategy
  - keyDecodingStrategy
  - nonConformingFloatDecodingStrategy

DECODING DATES FROM JSON

```
GET "https://example.com/movies/14"
{"id": 14, "title": "Star Wars IV: A New Hope", "director": "George Lucas",
"releaseDate": "1977-05-25"}

struct Movie: Codable {
    let id: Int
    let title: String
    let director: String
    let releaseDate: Date
}

let decoder = JSONDecoder()
let dateFormatter = DateFormatter()
dateFormatter.format = "yyyy-MM-dd"
decoder.dateDecodingStrategy = .formatted(dateFormatter)
let movie = try decoder.decode(Movie.self, from: data)
```

KEY DECODING STRATEGY

```
GET "https://example.com/movies/14"
{"id": 14, "title": "Star Wars IV: A New Hope", "director": "George Lucas",
"release_date": "1977-05-25"}

struct Movie: Codable {
    let id: Int
    let title: String
    let director: String
    let releaseDate: Date
}

let decoder = JSONDecoder()
let dateFormatter = ...
decoder.dateDecodingStrategy = .formatted(dateFormatter)
decoder.keyDecodingStrategy = .convertFromSnakeCase
let movie = try decoder.decode(Movie.self, from: data)
```

CODING KEYS

```
struct Movie: Codable {
    let id: Int
    let title: String
    let director: String
    let releaseDate: Date
}

extension Movie {
    enum CodingKeys: String, CodingKey {
        case id
        case title
        case director
        case releaseDate = "release_date"
    }
}
```

CUSTOM ENCODING

```
struct Movie: Codable {
    let id: Int
    let title: String
    let director: String
}

extension Movie {
    func encode(to encoder: Encoder) throws {
        var container = encoder.container(keyedBy: CodingKeys.self)
        try container.encode(id, forKey: .id)
        try container.encode(title, forKey: .title)
        try container.encode(director, forKey: .director)
    }
}
```

CUSTOM DECODING

```
struct Movie: Codable {
    let id: Int
    let title: String
    let director: String
}

extension Movie {
    init(from decoder: Decoder) throws {
        let container = try decoder.container(keyedBy: CodingKeys.self)
        let id = try container.decode(Int.self, forKey: .id)
        let title = try container.decode(String.self, forKey: .title)
        let director = try container.decode(String.self, forKey: .director)
        self.init(id: id, title: title, director: director)
    }
}
```

OTHER USAGES

OTHER FORMATS

- Property Lists (Apple)
- YAML (JP Simard)
- BSON (Kaitlin Mahar)
- Dictionary (Nicolas Zinovieff)
- XML (Frank Lefebvre)
- ...

TRANSCODING

```
struct Movie: Codable {
    ...
}

let decoder = JSONDecoder()
let encoder = PropertyListEncoder()

let jsonData = // JSON representation of movie
let movie = try decoder.decode(Movie.self, from: jsonData)
let plistData = try encoder.encode(movie)
```

CODABLE ROUTING (KITURA)

```
router.get("/movies", handler: getMovieHandler)
router.post("/movies", handler: postMovieHandler)

function getMovieHandler(id: Int, completion: (Movie?, RequestError?) -
> Void) -> Void {
    let movie = // retrieve from database with id
    completion(movie, nil)
}

function postMovieHandler(movie: Movie, completion: (Movie?,
RequestError?) -> Void) -> Void {
    movie.id = // create unique id
    // store movie into database
    completion(movie, nil)
}
```

DATABASE RECORDS (SWIFT-KUERY-ORM)

```
extension Movie: Model {}

let pool = PostgreSQLConnection.createPool(...)
Database.default = Database(pool)

try Movie.createTableSync()

let movie = Movie(...)
movie.save { movie, error in ... }

Movie.find(id: 14) { id, result, error in ... }
```

FURTHER DOCUMENTATION

- Ben Scheirman: http://swiftjson.guide
- Mike Ash: https://mikeash.com/pyblog/friday-qa-2017-07-14-swiftcodable.html
- John Sundell: https://www.swiftbysundell.com/articles/customizing-codable-types-in-swift/
- Chris Eidhof, Ole Begemann, Ben Cohen: Advanced Swift
- Mattt Zmuda: Flight School Guide to Swift Codable

OTHER ENCODER/DECODER IMPLEMENTATIONS

- YAML: https://github.com/jpsim/Yams.git
- BSON: https://github.com/mongodb/mongo-swift-driver
- Dictionary: https://github.com/krugazor/DictionaryCoding
- Binary: https://github.com/mikeash/BinaryCoder
- XML: https://github.com/franklefebvre/XMLCoder

QUESTIONS

@franklefebvre