Decodable vs real-world JSON

iOS-meetup SuperJob 30 ноября 2017

Владимир Бурдуков

NETCO

sports

Порядок повествования

1. Как было раньше?
2. Как стало сейчас?
3. Где это применить?

Наша песочница

```
struct Player {

   let firstName: String
   let lastName: String
   let displayName: String?
   let team: Team

}
```

Наша песочница

```
{
  "first_name": "Cristiano Ronaldo",
  "last_name": "dos Santos Aveiro",
  "display_name": "Cristiano Ronaldo",
  "team": {"name": "Portugal"}
}
```

Как было раньше

Из коробки

```
struct Player {
  let firstName: String, lastName: String, displayName: String?, team: Team

  init(_ json: [String: Any]) throws {
    guard let firstName = json["first_name"] as? String else { throw ... }
    self.firstName = firstName

    guard let lastName = json["last_name"] as? String else { throw ... }
    self.lastName = lastName

    displayName = json["display_name"] as? String

    guard let teamJSON = json["team"] as? [String: Any] else { throw ... }
    team = try Team(teamJSON)
  }
}
```

Как было раньше

SwiftyJSON

```
struct Player {
  let firstName: String, lastName: String, displayName: String?, team: Team

  init(_ json: JSON) throws {
    guard let firstName = json["first_name"].string else { throw ... }
    self.firstName = firstName

    guard let lastName = json["last_name"].string else { throw ... }
    self.lastName = lastName

    displayName = json["display_name"].string

    team = try Team(json["team"])
  }
}
```

Как было раньше

SwiftyJSON

```
struct Player {
  let firstName: String, lastName: String
  let displayName: String?, team: Team

  init(_ json: JSON) throws {
    self.firstName = json["first_name"].stringValue
    self.lastName = json["last_name"].stringValue
    displayName = json["display_name"].string
    team = try Team(json["team"])
  }
}
```

Как было раньше

Argo

```
struct Player {
  let firstName: String, lastName: String, displayName: String?, team: Team

  static func decode(_ json: JSON) -> Decoded<Player> {
    return curry(Player.init)
      <^> json <| "first_name"
      <*> json <| "last_name"
      <*> json <|? "display_name"
      <*> json <| "team"
  }
}
```

Как было раньше

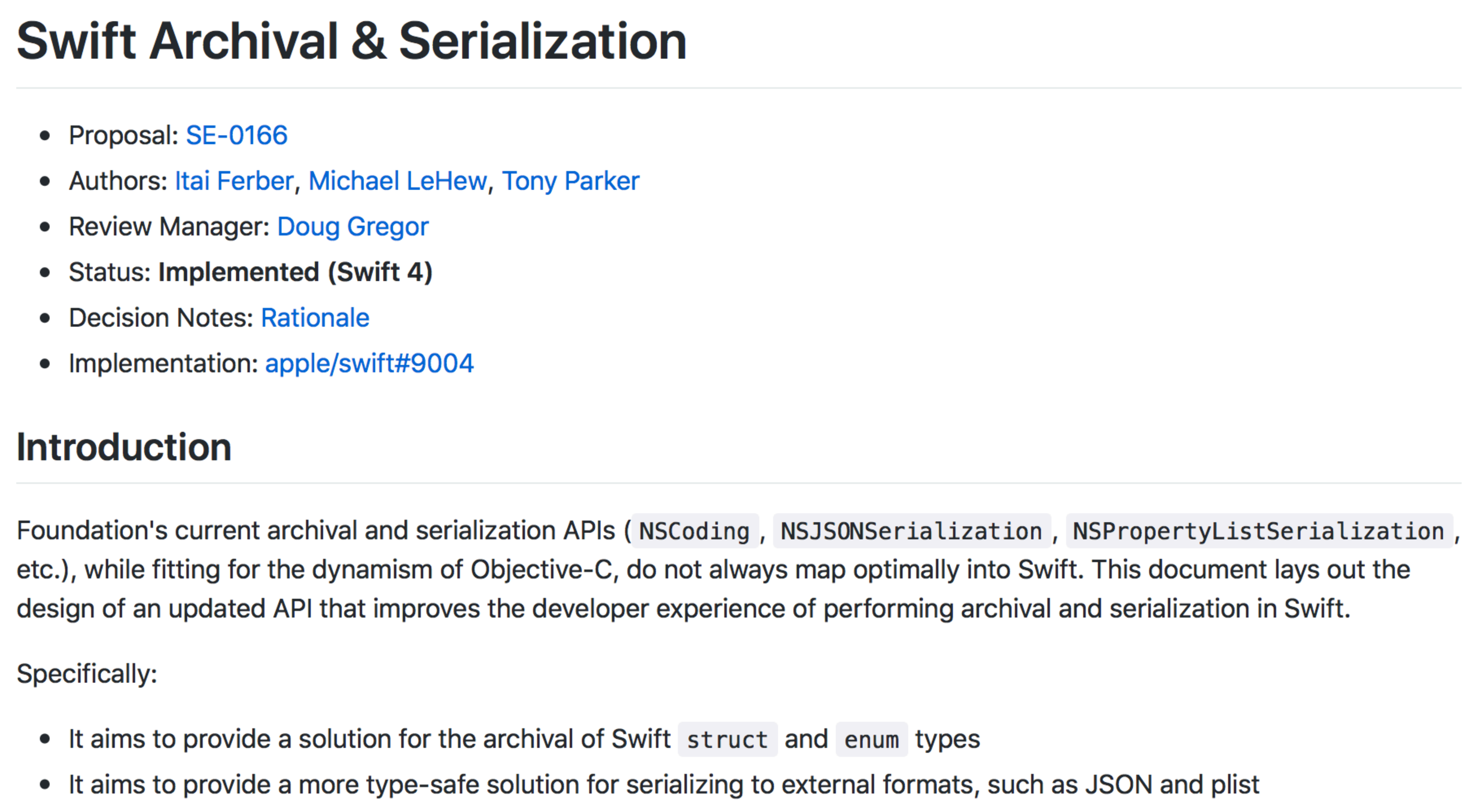
Вставьте название своей любимой библиотеки

Как стало сейчас

```swift
struct Player: Decodable {
  let firstName: String, lastName: String
  let displayName: String?, team: Team

  private enum CodingKeys: String, CodingKey {
    case firstName = "first_name"
    case lastName = "last_name"
    case displayName = "display_name", team
  }
}

let decoder = JSONDecoder()
let player = try decoder.decode(Player.self, from: data)
```



Swift Archival & Serialization

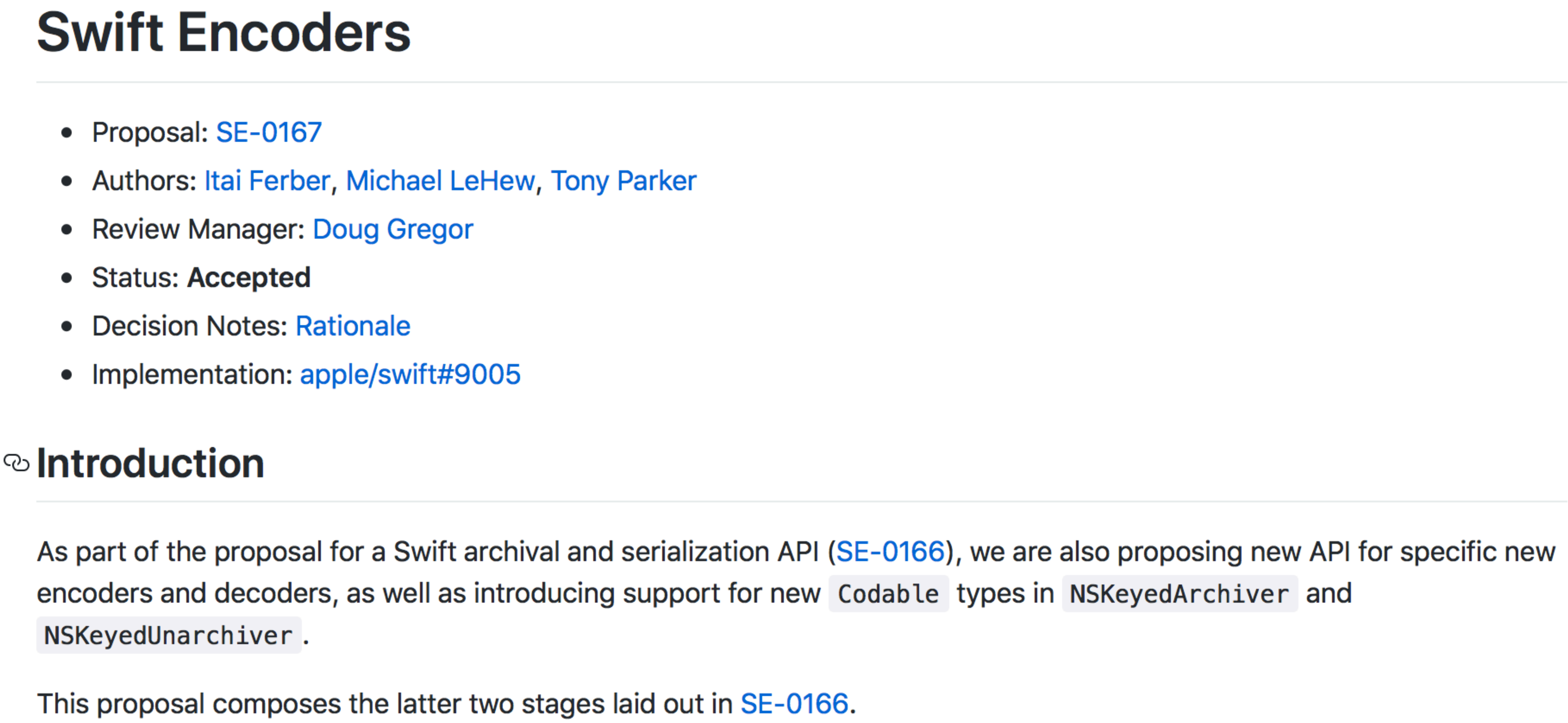
- Proposal: SE-0166
- Authors: Itai Ferber, Michael LeHew, Tony Parker
- Review Manager: Doug Gregor
- Status: **Implemented (Swift 4)**
- Decision Notes: Rationale
- Implementation: apple/swift#9004

Introduction

Foundation's current archival and serialization APIs (`NSCoding`, `NSJSONSerialization`, `NSPropertyListSerialization`, etc.), while fitting for the dynamism of Objective-C, do not always map optimally into Swift. This document lays out the design of an updated API that improves the developer experience of performing archival and serialization in Swift.

Specifically:

- It aims to provide a solution for the archival of Swift `struct` and `enum` types
- It aims to provide a more type-safe solution for serializing to external formats, such as JSON and plist



Swift Encoders

- Proposal: SE-0167
- Authors: Itai Ferber, Michael LeHew, Tony Parker
- Review Manager: Doug Gregor
- Status: **Accepted**
- Decision Notes: Rationale
- Implementation: apple/swift#9005

Introduction

As part of the proposal for a Swift archival and serialization API (SE-0166), we are also proposing new API for specific new encoders and decoders, as well as introducing support for new `Codable` types in `NSKeyedArchiver` and `NSKeyedUnarchiver`.

This proposal composes the latter two stages laid out in SE-0166.

Как стало сейчас

```
typealias Codable = Encodable & Decodable
```

Как стало сейчас

```
public protocol Decodable {
  init(from decoder: Decoder) throws
}
```

Как стало сейчас

В чём польза?

```
struct Player: Decodable {
  let firstName: String, lastName: String
  let displayName: String?, team: Team

  private enum CodingKeys: String, CodingKey {
    case firstName = "first_name"
    case lastName = "last_name"
    case displayName = "display_name", team
  }
}
```

Как стало сейчас

В чём польза?

1. Для простых моделей генерируется код при компиляции

```swift
struct Player: Decodable {
  let firstName: String, lastName: String
}
```

```
struct Player: Decodable {
  let firstName: String, lastName: String
  @derived enum CodingKeys: String, CodingKey {
    case firstName, lastName
  }

  @derived init(from decoder: Decoder) throws {
    let container = decoder.container(keyedBy: CodingKeys.self)
    firstName = try container.decode(String.self,
                                     forKey: .firstName)
    lastName = try container.decode(String.self,
                                    forKey: .lastName)
  }
}
```

```
struct Player: Decodable {
  let firstName: String, lastName: String
  enum CodingKeys: String, CodingKey {
    case firstName = "first_name", lastName = "last_name"
  }

  @derived init(from decoder: Decoder) throws {
    let container = decoder.container(keyedBy: CodingKeys.self)
    firstName = try container.decode(String.self,
                                     forKey: .firstName)
    lastName = try container.decode(String.self,
                                    forKey: .lastName)
  }
}
```

Как стало сейчас

Что внутри?

Как стало сейчас

Decoder

```
public protocol Decoder {
  var codingPath: [CodingKey] { get }
  var userInfo: [CodingUserInfoKey : Any] { get }

  func container<Key>(keyedBy type: Key.Type) throws
                      -> KeyedDecodingContainer<Key>
  func unkeyedContainer() throws -> UnkeyedDecodingContainer
  func singleValueContainer() throws -> SingleValueDecodingContainer
}
```

Как стало сейчас

KeyedDecodingContainer

```
public struct KeyedDecodingContainer<Key: CodingKey> {
  var codingPath: [CodingKey]
  var allKeys: [Key]
  func contains(_ key: Key) -> Bool
  ...
}
```

Как стало сейчас

KeyedDecodingContainer

```
public struct KeyedDecodingContainer<Key: CodingKey> {
  ...
  func decodeNil(forKey key: Key) throws -> Bool
  // decode Int, String, Bool, etc.
  func decode<T : Decodable>(_ type: T.Type,
                             forKey key: Key) throws -> T
  // decodeIfPresent Int, String, Bool, etc.
  func decodeIfPresent<T : Decodable>(_ type: T.Type,
                                      forKey key: Key) throws -> T?
  ...
}
```

Как стало сейчас

KeyedDecodingContainer

```
public struct KeyedDecodingContainer<Key: CodingKey> {
  ...
  func nestedContainer<NestedKey>(keyedBy type: NestedKey.Type,
                                  forKey key: Key) throws
                                  -> KeyedDecodingContainer<NestedKey>
  func nestedUnkeyedContainer(forKey key: Key) throws -> UnkeyedDecodingContainer
  func superDecoder() throws -> Decoder
  func superDecoder(forKey key: Key) throws -> Decoder
}
```

Как стало сейчас

UnkeyedDecodingContainer

```
public protocol UnkeyedDecodingContainer {
  var codingPath: [CodingKey] { get }
  var count: Int? { get }
  var isAtEnd: Bool { get }
  var currentIndex: Int { get }
}
```

Как стало сейчас

UnkeyedDecodingContainer

```
public protocol UnkeyedDecodingContainer {
  ...
  mutating func decodeNil() throws -> Bool
  // decode Int, String, Bool, etc.
  mutating func decode<T : Decodable>(_ type: T.Type) throws -> T
  // decodeIfPresent Int, String, Bool, etc.
  mutating func decodeIfPresent<T : Decodable>(_ type: T.Type) throws -> T?
  ...
}
```

Как стало сейчас

UnkeyedDecodingContainer

```
public protocol UnkeyedDecodingContainer {
  ...
  mutating func nestedContainer<NestedKey>(keyedBy type: NestedKey.Type) throws
                                           -> KeyedDecodingContainer<NestedKey>
  mutating func nestedUnkeyedContainer() throws -> UnkeyedDecodingContainer
  mutating func superDecoder() throws -> Decoder
}
```

Как стало сейчас

SingleValueDecodingContainer

```
public protocol SingleValueDecodingContainer {
  var codingPath: [CodingKey] { get }
  func decodeNil() -> Bool
  func decode<T : Decodable>(_ type: T.Type) throws -> T
}
```

Как стало сейчас

Стандартные типы

- Int, Bool, String, Double ...
- Array, Set, Dictionary ...
- некоторые типы Foundation и CoreGraphics

Как стало сейчас

Реализации

1. JSONDecoder
2. PropertyListDecoder
3. NSKeyedUnarchiver (?)

Как стало сейчас

JSONDecoder

1. принимает на вход тип, реализующий Decodable, и Data
2. можно конфигурировать:
   - как парсить Date
   - как парсить Data
   - как парсить Float
3. можно добавлять userInfo, который может

Как стало сейчас

JSONDecoder

```
let decoder = JSONDecoder()
decoder.dataDecodingStrategy = .base64
decoder.dateDecodingStrategy = .secondsSince1970
let player = try decoder.decode(Player.self, from: data)
```

Где это применить?

Где это применить?

Пиар

Gnomon

Astrolabe

Live Coding

Playground лежит на Github