

**Macoun**

# Think Codables

Christopher Beloch - @SomeKindOfCode

# Ablauf

- Was sind Codables?
- (De-)Serialisierung anpassen
- Arbeiten mit Generics
- 3rd Party

**Was sind Codables?**

# Was sind Codables?

- zentrale Protokolle
- Data nach/von Model

# JSON Library

```
init(json: JSON) {  
    self.firstName = json["first_name"].string  
    self.lastName = json["last_name"].string  
    self.session = json["session_title"].string  
}
```

# JSON Library

```
init(json: JSON) {  
    self.firstName = json["first_name"].string  
    self.lastName = json["last_name"].string  
    self.session = json["session_title"].string  
}
```

# JSONSerialization

```
init(dict: [String: Any]) {  
    self.firstName = dict["first_name"] as? String  
    self.lastName = dict["last_name"] as? String  
    self.session = dict["session_title"] as? String  
}
```



# Protokolle

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

# Protokolle

```
typealias Codable = Decodable & Encodable  
  
// Decodable  
init(from decoder: Decoder) throws
```

# Protokolle

```
typealias Codable = Decodable & Encodable

// Decodable
init(from decoder: Decoder) throws

// Encodable
func encode(to encoder: Encoder) throws
```

# Objekt Codable machen

```
struct Session {  
    let title: String  
    let from: Date  
    let duration: TimeInterval  
    let details: URL  
}
```

# Objekt Codable machen

```
struct Session {  
    let title: String  
    let from: Date  
    let duration: TimeInterval  
    let details: URL  
}  
  
extension Session: Codable {}
```

# Existierende Codables

String

Int / Float / Double

Bool

Data

Date

URL

Array / Set

Dictionary

Enum

CGPoint

CGSize

CGRect

UUID

Locale

IndexPath

ClosedRange

**(De-)Serialisierung anpassen**

# (De-)Serialisierung anpassen

- De-/Encoder Optionen
- CodingKeys
- Initialiser oder encode Methode



# De-/Encoder Optionen

# PropertyList Encoder

```
var encoder: PropertyListEncoder = .init()  
encoder.outputFormat = .binary
```

# PropertyList Encoder

```
var encoder: PropertyListEncoder = .init()  
encoder.outputFormat = .xml
```

# PropertyList Encoder

```
var encoder: PropertyListEncoder = .init()  
encoder.outputFormat = .openStep
```

# PropertyList Encoder

```
var encoder: PropertyListEncoder = .init()
encoder.outputFormat = .openStep

let decoder: PropertyListDecoder = .init()

// no options
```

# JSONEncoder

```
var encoder: JSONEncoder = .init()  
encoder.outputFormatting = .prettyPrinted
```

# JSONEncoder

```
var encoder: JSONEncoder = .init()  
encoder.outputFormatting = .sortedKeys
```

# JSONEncoder

```
var encoder: JSONEncoder = .init()  
encoder.outputFormatting = [.prettyPrinted, .sortedKeys]
```



# JSONEncoder

```
var encoder: JSONEncoder = .init()

encoder.outputFormatting = [.prettyPrinted, .sortedKeys]
encoder.keyEncodingStrategy = .convertToCamelCase
```

# JSONEncoder

```
var encoder: JSONEncoder = .init()

encoder.outputFormatting = [.prettyPrinted, .sortedKeys]
encoder.keyEncodingStrategy = .convertToCamelCase
encoder.dataEncodingStrategy = .base64
```

# JSONEncoder

```
var encoder: JSONEncoder = .init()

encoder.outputFormatting = [.prettyPrinted, .sortedKeys]
encoder.keyEncodingStrategy = .convertToCamelCase
encoder.dataEncodingStrategy = .base64
encoder.dateEncodingStrategy = .iso8601
```

# JSONEncoder

```
var encoder: JSONEncoder = .init()

encoder.outputFormatting = [.prettyPrinted, .sortedKeys]
encoder.keyEncodingStrategy = .convertToCamelCase
encoder.dataEncodingStrategy = .base64
encoder.dateEncodingStrategy = .formatted(formatter)
```

# JSONEncoder

```
var encoder: JSONEncoder = .init()

encoder.outputFormatting = [.prettyPrinted, .sortedKeys]
encoder.keyEncodingStrategy = .convertToCamelCase
encoder.dataEncodingStrategy = .base64
encoder.dateEncodingStrategy = .formatted(formatter)
encoder.nonConformingFloatEncodingStrategy = .throw
```

# JSONDecoder

```
var decoder: JSONDecoder = .init()

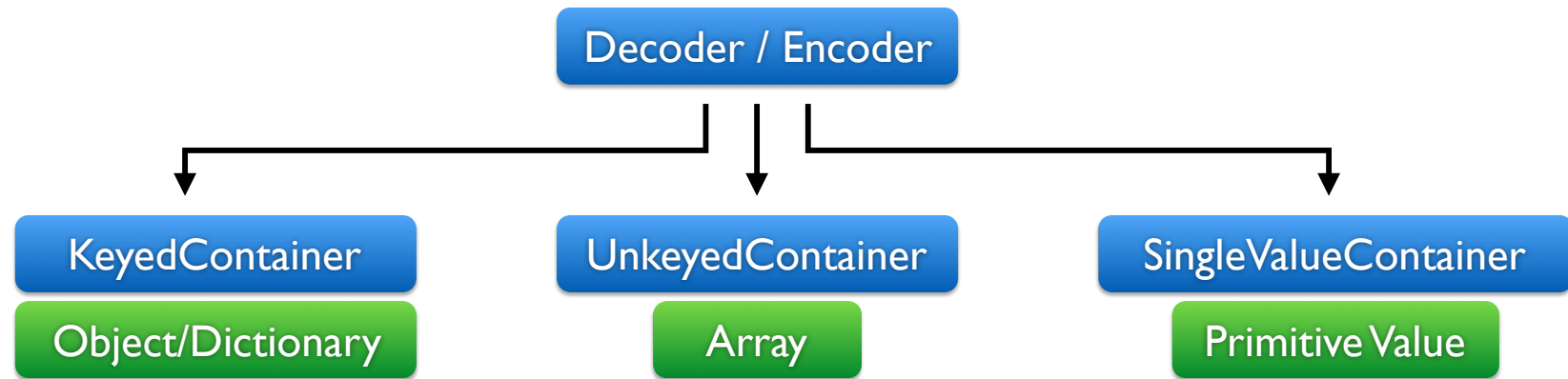
decoder.keyDecodingStrategy = .convertFromSnakeCase
decoder.dataDecodingStrategy = .base64
decoder.dateDecodingStrategy = .formatted(formatter)
decoder.nonConformingFloatDecodingStrategy = .throw
```

Demo

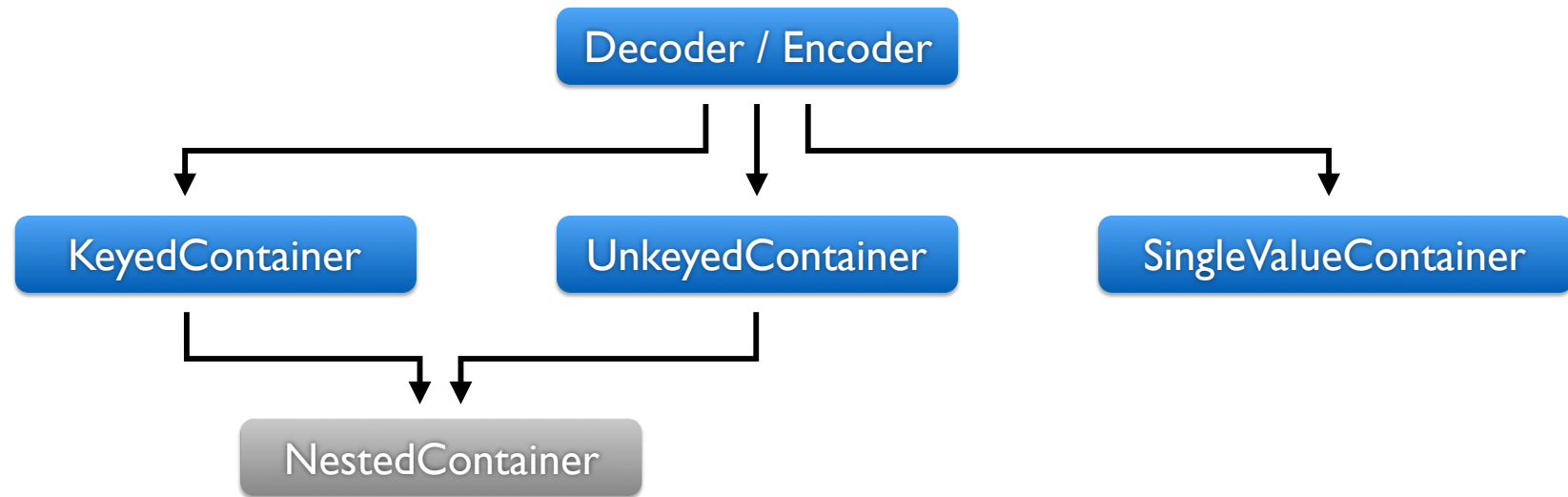
De-/EncodingContainer



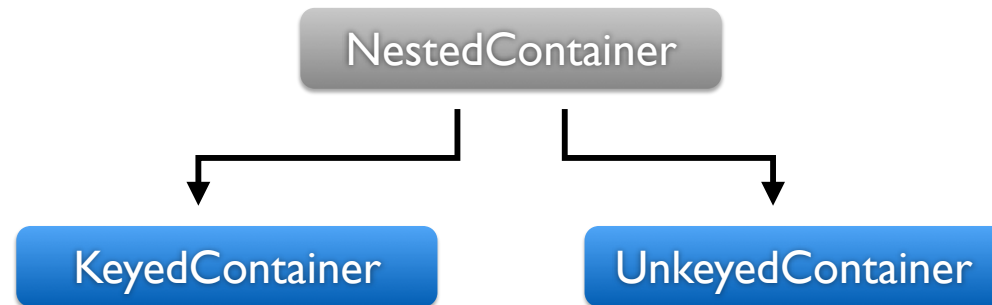
# De-/EncodingContainer



# De-/EncodingContainer






# De-/EncodingContainer



# Codable Mythen

# Codable Mythen

- JSONEncoder: Encoder 
- Optional Enum ist sicher 
- Jeder Datentyp ist wichtig 

# Generics und Mixed Arrays

# Generics und Mixed Arrays

- Proxy Objekt
- Codable konform
- ggf. eigenes Protokoll

Demo



**3rd Party**

# XMLCoder

Max Desiatov

[github.com/MaxDesiatov/XMLCoder](https://github.com/MaxDesiatov/XMLCoder)

# MessagePacker

hiro

[github.com/hirotakan/MessagePacker](https://github.com/hirotakan/MessagePacker)

# ZippyJSON

Michael Eisel

[github.com/michael-eisel/ZippyJSON](https://github.com/michael-eisel/ZippyJSON)

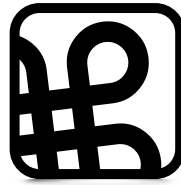
# Codextended

John Sundell

[github.com/JohnSundell/Codextended](https://github.com/JohnSundell/Codextended)

Fragen?

**Vielen Dank**



**Macoun**