Encoding and Decoding

Tony Parker, Foundation

Encoding and Decoding

Conversion between Swift data structures and archived formats

Encoding and Decoding

Conversion between Swift data structures and archived formats

Swift and archived formats have strong typing mismatch

Encoding and Decoding

Conversion between Swift data structures and archived formats

Swift and archived formats have strong typing mismatch

Solution is close integration with Swift

```
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
```

```json
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
```

```swift
struct Author {
  let name: String
  let email: String
  let date: Date
}
```

```
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
```

```
struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}
```

```swift
let jsonData = """
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
""".data(using: .utf8)!

struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}
```

```swift
let jsonData = """
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
""".data(using: .utf8)!

struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}

let decoder = JSONDecoder()
let jsonData = """
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
""".data(using: .utf8)!

struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}

let decoder = JSONDecoder()
let jsonData = """
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
""".data(using: .utf8)!

struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}

let decoder = JSONDecoder()
decoder.dateDecodingStrategy = .iso8601
```

```swift
let jsonData = """
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
""".data(using: .utf8)!

struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}

let decoder = JSONDecoder()
decoder.dateDecodingStrategy = .iso8601
```

```swift
let jsonData = """
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
""".data(using: .utf8)!

struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}

let decoder = JSONDecoder()
decoder.dateDecodingStrategy = .iso8601
let author = try decoder.decode(Author.self, from: jsonData)
let jsonData = """
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
""".data(using: .utf8)!

struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}

let decoder = JSONDecoder()
decoder.dateDecodingStrategy = .iso8601
let author = try decoder.decode(Author.self, from: jsonData)
```

```
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
```

```json
{
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
```

```
          {
  "name": "Monalisa Octocat",
  "email": "support@github.com",
  "date": "2011-04-14T16:00:49Z"
}
```

```json
{
  "url": "https://api.github.com/.../6dcb09",
  "author": {
    "name": "Monalisa Octocat",
    "email": "support@github.com",
    "date": "2011-04-14T16:00:49Z"
  },
  "message": "Fix all the bugs",
  "comment_count": 0,
}
```

```
{
  "url": "https://api.github.com/.../6dcb09",
  "author": {
    "name": "Monalisa Octocat",
    "email": "support@github.com",
    "date": "2011-04-14T16:00:49Z"
  },
  "message": "Fix all the bugs",
  "comment_count": 0,
}
```

```swift
struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}
```

```json
{
  "url": "https://api.github.com/.../6dcb09",
  "author": {
    "name": "Monalisa Octocat",
    "email": "support@github.com",
    "date": "2011-04-14T16:00:49Z"
  },
  "message": "Fix all the bugs",
  "comment_count": 0,
}
```

```swift
struct Author : Codable {
  let name: String
  let email: String
  let date: Date
}
```

```json
{
  "url": "https://api.github.com/.../6dcb09",
  "author": {
    "name": "Monalisa Octocat",
    "email": "support@github.com",
    "date": "2011-04-14T16:00:49Z"
  },
  "message": "Fix all the bugs",
  "comment_count": 0,
}
```

```swift
struct Commit : Codable {
  let url: URL
  struct Author : Codable {
    let name: String
    let email: String
    let date: Date
  }
  let author: Author
  let message: String
  let comment_count: Int
}
```

```json
{
  "url": "https://api.github.com/.../6dcb09",
  "author": {
    "name": "Monalisa Octocat",
    "email": "support@github.com",
    "date": "2011-04-14T16:00:49Z"
  },
  "message": "Fix all the bugs",
  "comment_count": 0,
}
```

```swift
struct Commit : Codable {
  let url: URL
  struct Author : Codable {
    let name: String
    let email: String
    let date: Date
  }
  let author: Author
  let message: String
  let comment_count: Int
}
```

```swift
let commit = try decoder.decode(Commit.self, from: jsonData)
```

```json
{
  "url": "https://api.github.com/.../6dcb09",
  "author": {
    "name": "Monalisa Octocat",
    "email": "support@github.com",
    "date": "2011-04-14T16:00:49Z"
  },
  "message": "Fix all the bugs",
  "comment_count": 0,
}
```

```swift
struct Commit : Codable {
  let url: URL
  struct Author : Codable {
    let name: String
    let email: String
    let date: Date
  }
  let author: Author
  let message: String
  let comment_count: Int
}
```

```swift
let commit = try decoder.decode(Commit.self, from: jsonData)
```

```
{
  "url": "https://api.github.com/.../6dcb09",
  "author": {
    "name": "Monalisa Octocat",
    "email": "support@github.com",
    "date": "2011-04-14T16:00:49Z"
  },
  "message": "Fix all the bugs",
  "comment_count": 0,
}
```

```swift
struct Commit : Codable {
  let url: URL
  struct Author : Codable {
    let name: String
    let email: String
    let date: Date
  }
  let author: Author
  let message: String
  let comment_count: Int
}
```

```swift
let commit = try decoder.decode(Commit.self, from: jsonData)
let commitDate = commit.author.date
```

```json
{
  "url": "https://api.github.com/.../6dcb09",
  "author": {
    "name": "Monalisa Octocat",
    "email": "support@github.com",
    "date": "2011-04-14T16:00:49Z"
  },
  "message": "Fix all the bugs",
  "comment_count": 0,
}
```

```swift
struct Commit : Codable {
  let url: URL
  struct Author : Codable {
    let name: String
    let email: String
    let date: Date
  }
  let author: Author
  let message: String
  let comment_count: Int
}
```

```swift
let commit = try decoder.decode(Commit.self, from: jsonData)
let commitDate = commit.author.date
```

Coding Protocols

Codable

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

Coding Protocols

Codable

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

Encodable

```swift
public protocol Encodable {
  func encode(to encoder: Encoder) throws
}
```

Coding Protocols

Codable

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

Encodable

```
public protocol Encodable {
  func encode(to encoder: Encoder) throws
}
```

Decodable

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

Coding Protocols

Use Swift protocol extension behavior

Coding Protocols

Use Swift protocol extension behavior

Write your own implementation to customize

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int

  // Encodable
  public func encode(to encoder: Encoder) throws { /* … */ }

  // Decodable
  init(from decoder: Decoder) throws { /* … */ }
```

Compiler Generated

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int

  private enum CodingKeys : String, CodingKey {
    case url
    case message
    case author
    case comment_count
  }
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int

  private enum CodingKeys : String, CodingKey {
    case url
    case message
    case author
    case comment_count
  }
```

Compiler Generated

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int

  private enum CodingKeys : String, CodingKey {
    case url
    case message
    case author
    case comment_count
  }
```

Customized

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int

  private enum CodingKeys : String, CodingKey {
    case url
    case message
    case author
    case comment_count
  }
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let comment_count: Int

  private enum CodingKeys : String, CodingKey {
    case url
    case message
    case author
    case comment_count
  }
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int

  private enum CodingKeys : String, CodingKey {
    case url
    case message
    case author
    case commentCount = "comment_count"
  }
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int

  private enum CodingKeys : String, CodingKey {
    case url
    case message
    case author
    case commentCount = "comment_count"
  }
}
```

Demo

Encoding and Decoding

Itai Ferber, Foundation

Encoding and Decoding

Tony Parker, Foundation

Codable Philosophy

Error handling built-in

Encapsulate encoding details

Abstract format from types

Codable Philosophy

Error handling built-in

Encapsulate encoding details

Abstract format from types

Error Handling

Unexpected input is not if, but when

Error Handling

Unexpected input is not if, but when

No fatal errors from untrusted data—only for developer mistakes

Error Handling

Unexpected input is not if, but when

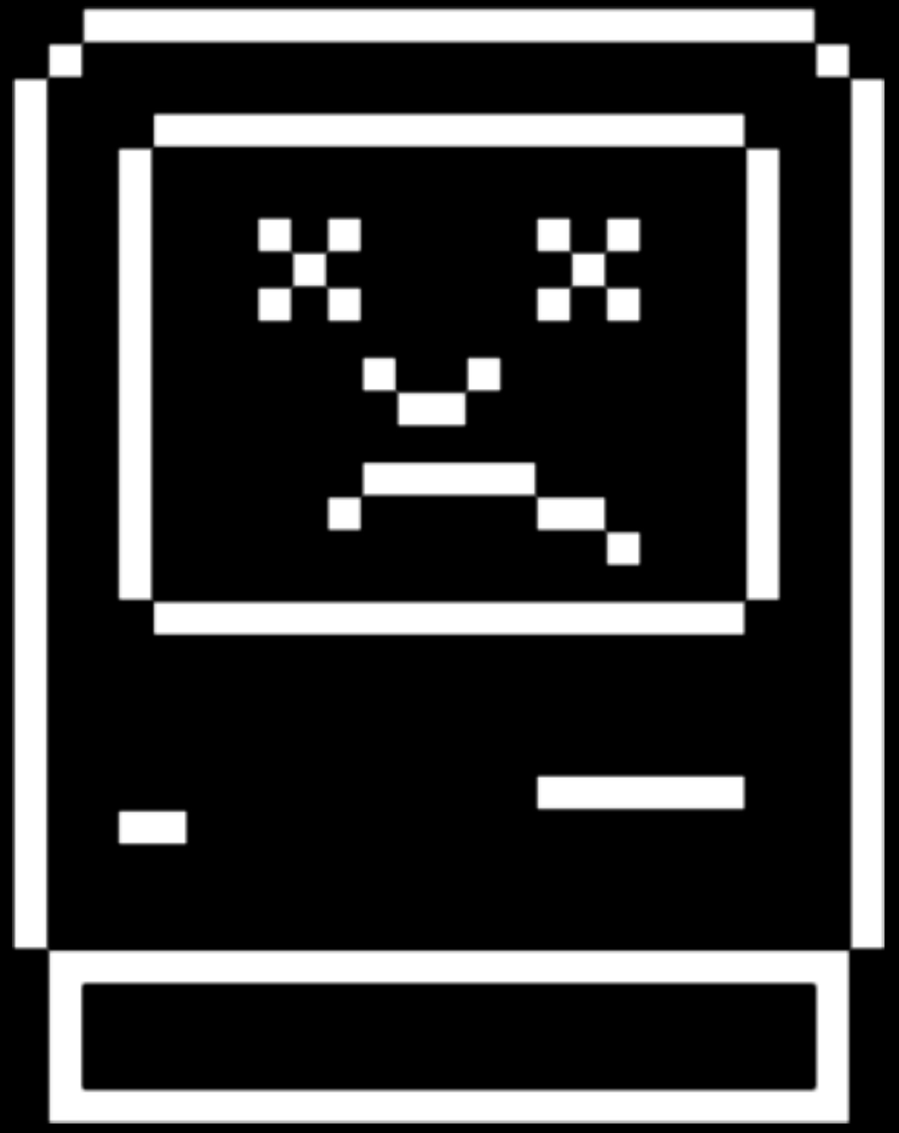
No fatal errors from untrusted data—only for developer mistakes

Errors possible on decode and encode

Coder Errors

Encoding

- Invalid value



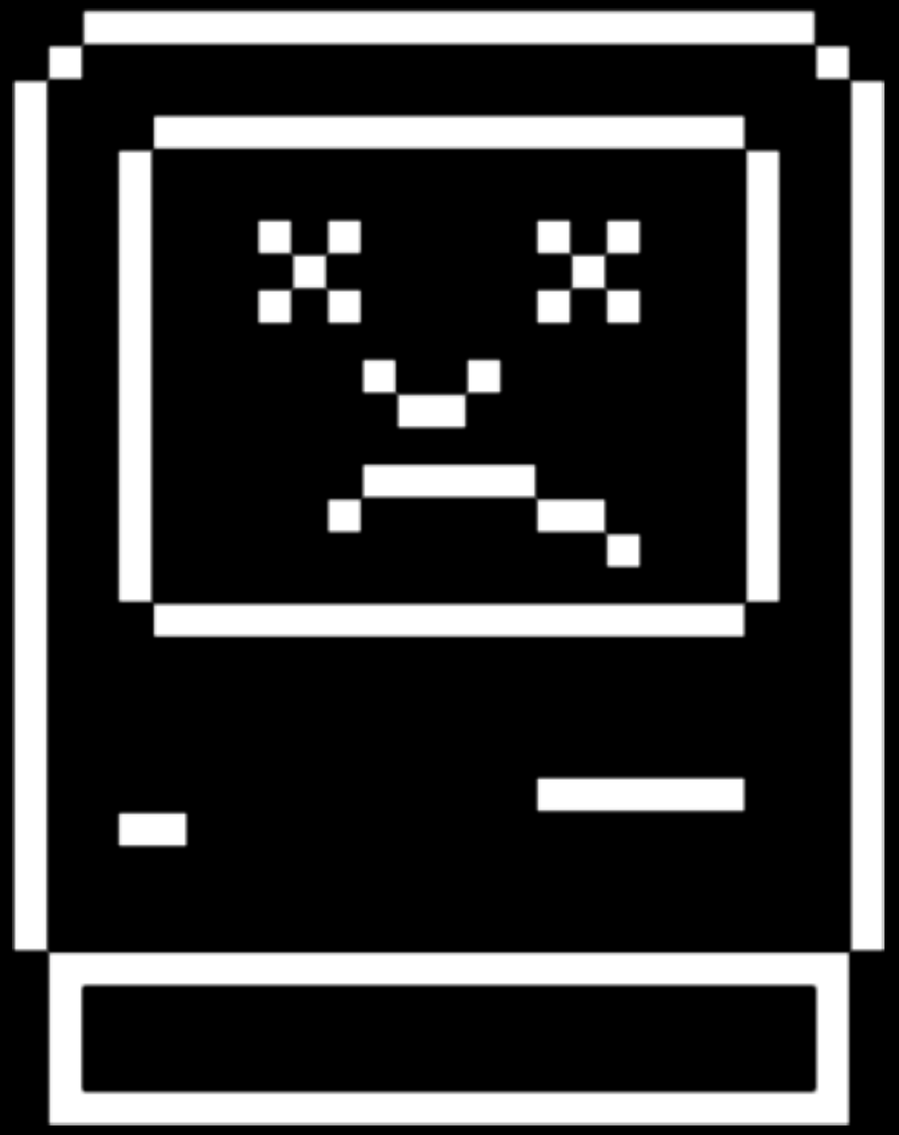
Coder Errors

Encoding

- Invalid value

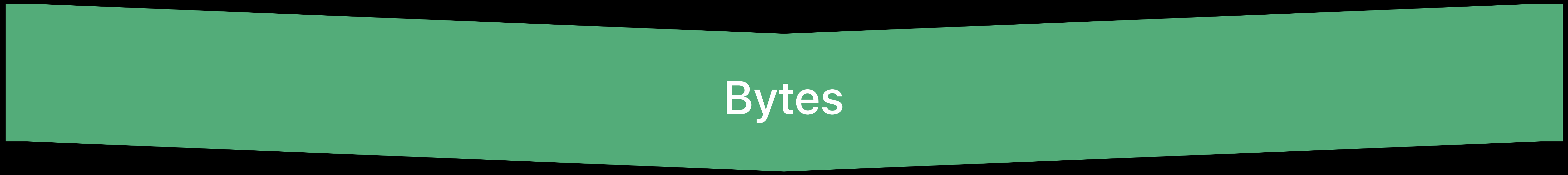
Decoding

- Type mismatch
- Missing key
- Missing value
- Data corrupt

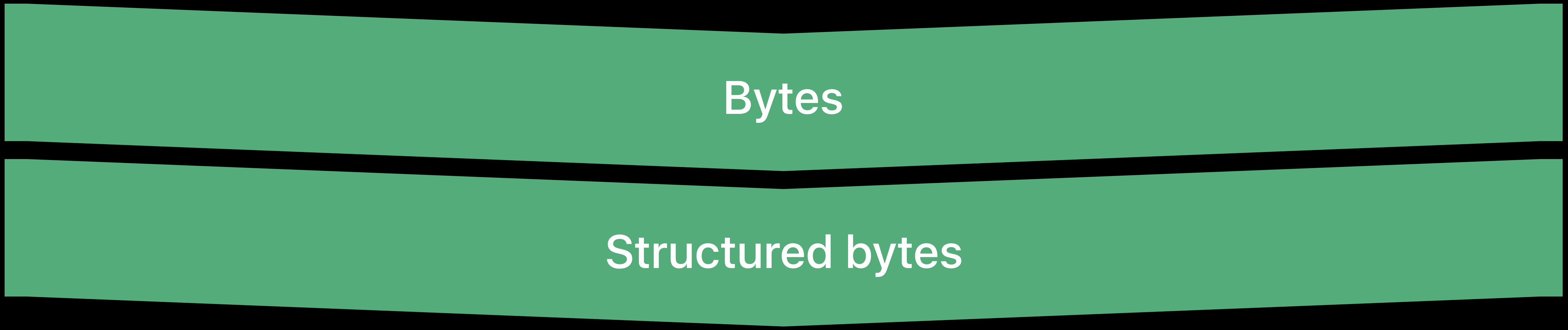


Beyond Basic Error Handling

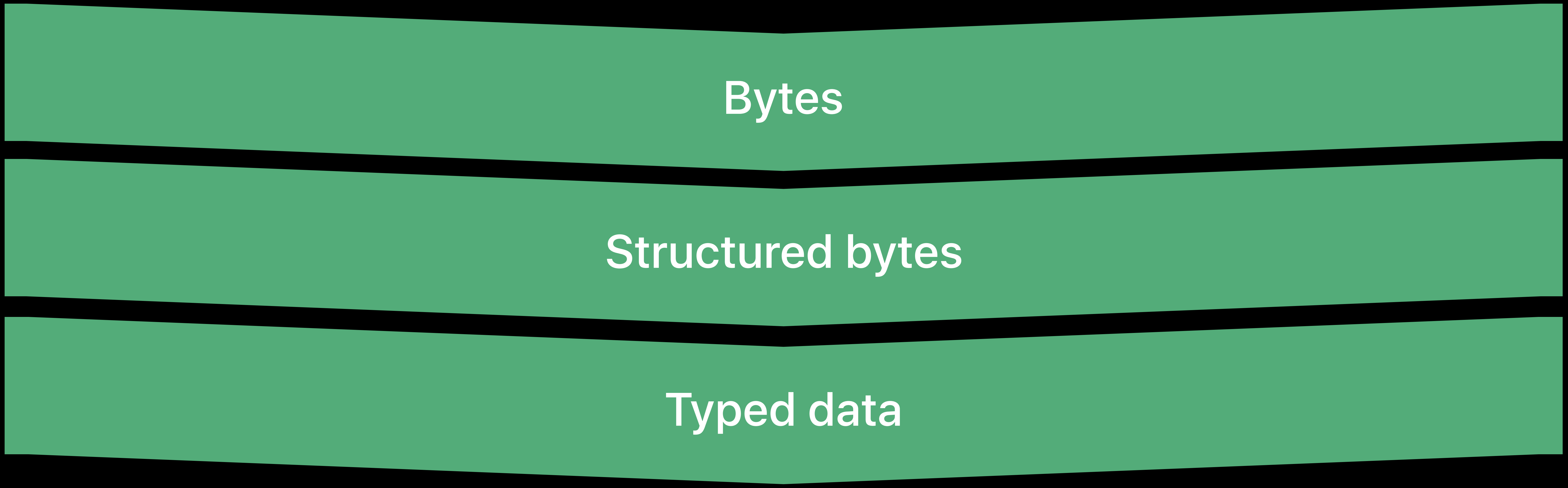
Beyond Basic Error Handling



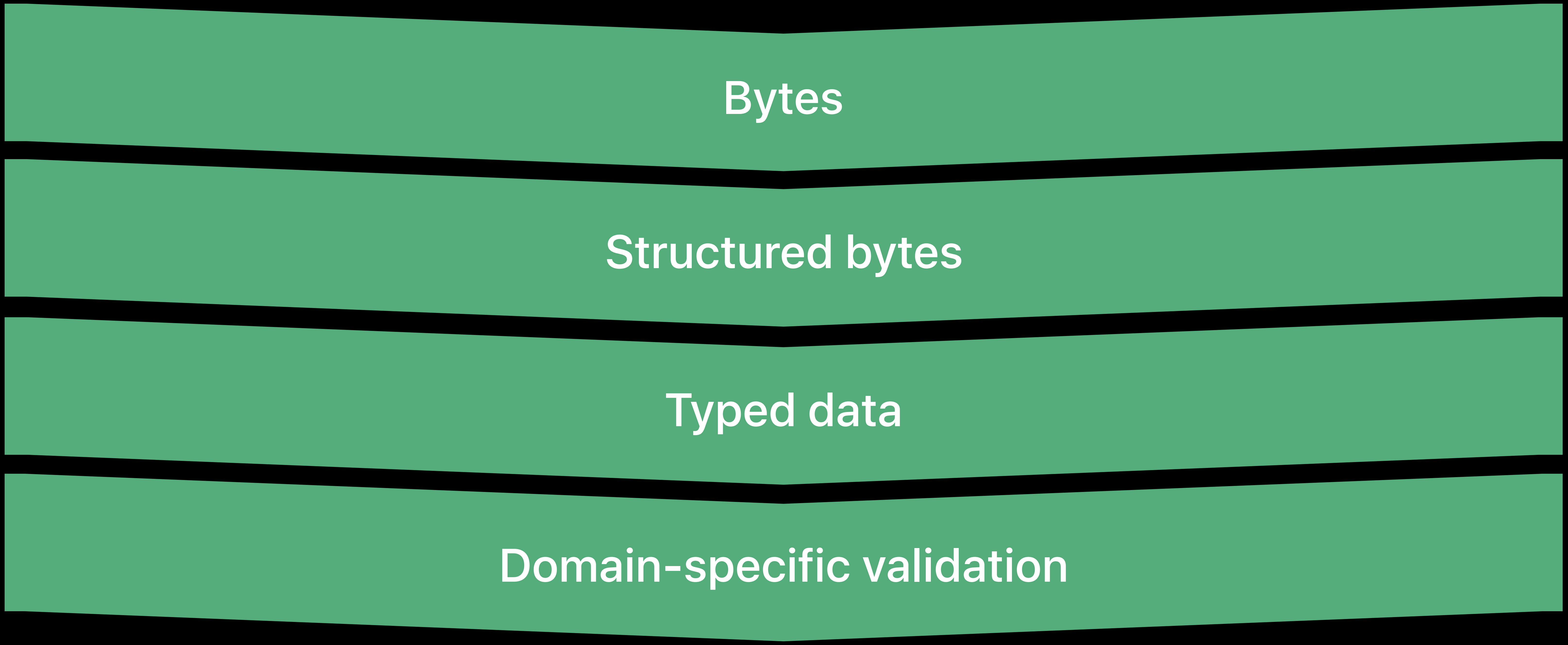
Beyond Basic Error Handling



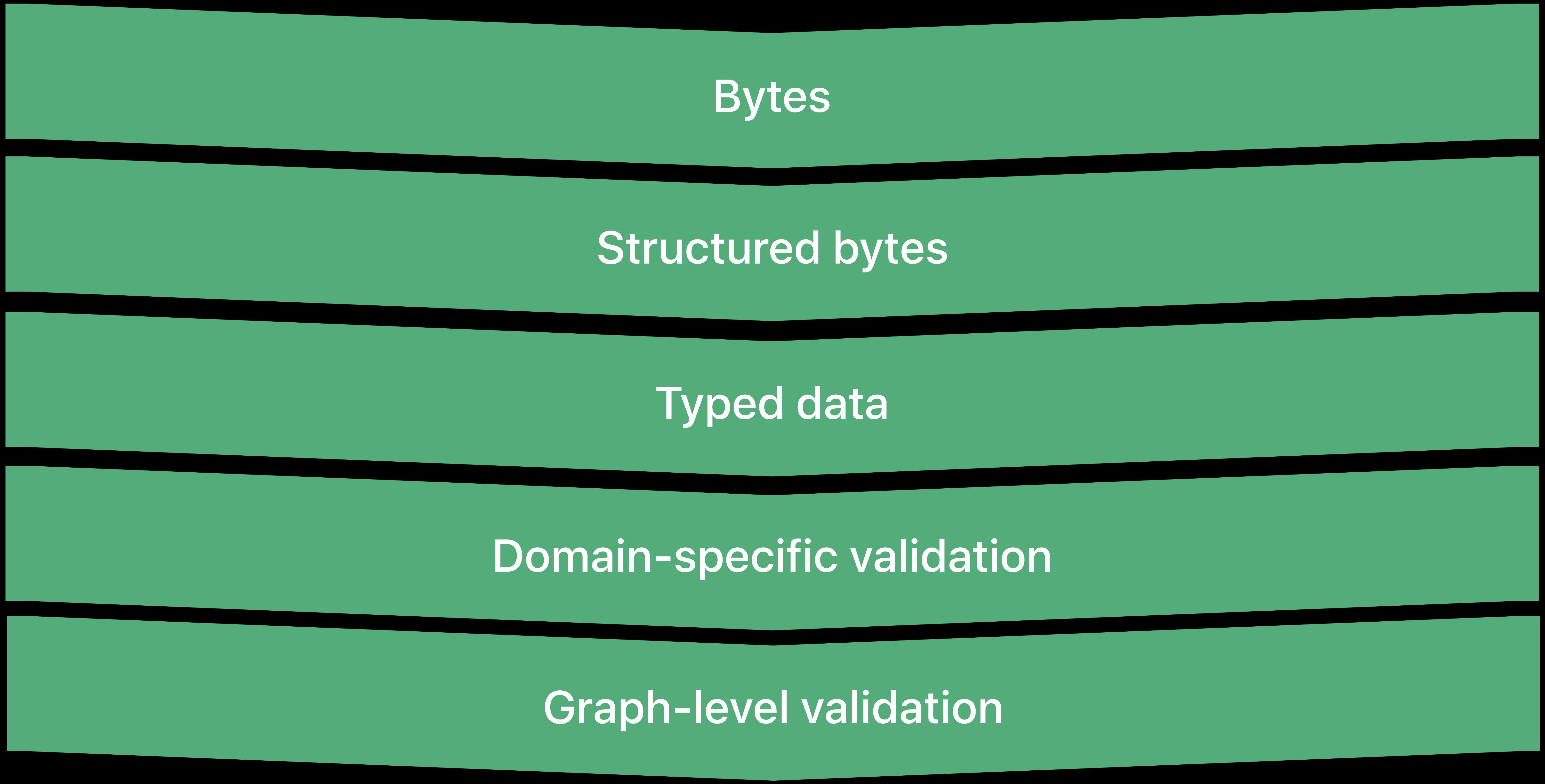
Beyond Basic Error Handling



Beyond Basic Error Handling



Beyond Basic Error Handling



```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    url = try container.decode(URL.self, forKey: .url)
    message = try container.decode(String.self, forKey: .message)
    author = try container.decode(Author.self, forKey: .author)
    commentCount = try container.decode(Int.self, forKey: .commentCount)
  }
}
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    url = try container.decode(URL.self, forKey: .url)
    message = try container.decode(String.self, forKey: .message)
    author = try container.decode(Author.self, forKey: .author)
    commentCount = try container.decode(Int.self, forKey: .commentCount)
  }
}
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    url = try container.decode(URL.self, forKey: .url)
    message = try container.decode(String.self, forKey: .message)
    author = try container.decode(Author.self, forKey: .author)
    commentCount = try container.decode(Int.self, forKey: .commentCount)
  }
}
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    url = try container.decode(URL.self, forKey: .url)
    message = try container.decode(String.self, forKey: .message)
    author = try container.decode(Author.self, forKey: .author)
    commentCount = try container.decode(Int.self, forKey: .commentCount)
  }
}
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    url = try container.decode(URL.self, forKey: .url)





    message = try container.decode(String.self, forKey: .message)
    author = try container.decode(Author.self, forKey: .author)
    commentCount = try container.decode(Int.self, forKey: .commentCount)
  }
}
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    url = try container.decode(URL.self, forKey: .url)
    guard url.scheme == "https" else {
      throw DecodingError.dataCorrupted(DecodingError.Context(
              codingPath: container.codingPath + [CodingKeys.url],
              debugDescription: "URLs require https")) }
    message = try container.decode(String.self, forKey: .message)
    author = try container.decode(Author.self, forKey: .author)
    commentCount = try container.decode(Int.self, forKey: .commentCount)
  }
}
```

Codable Philosophy

Error handling built-in

Encapsulate encoding details

Abstract format from types

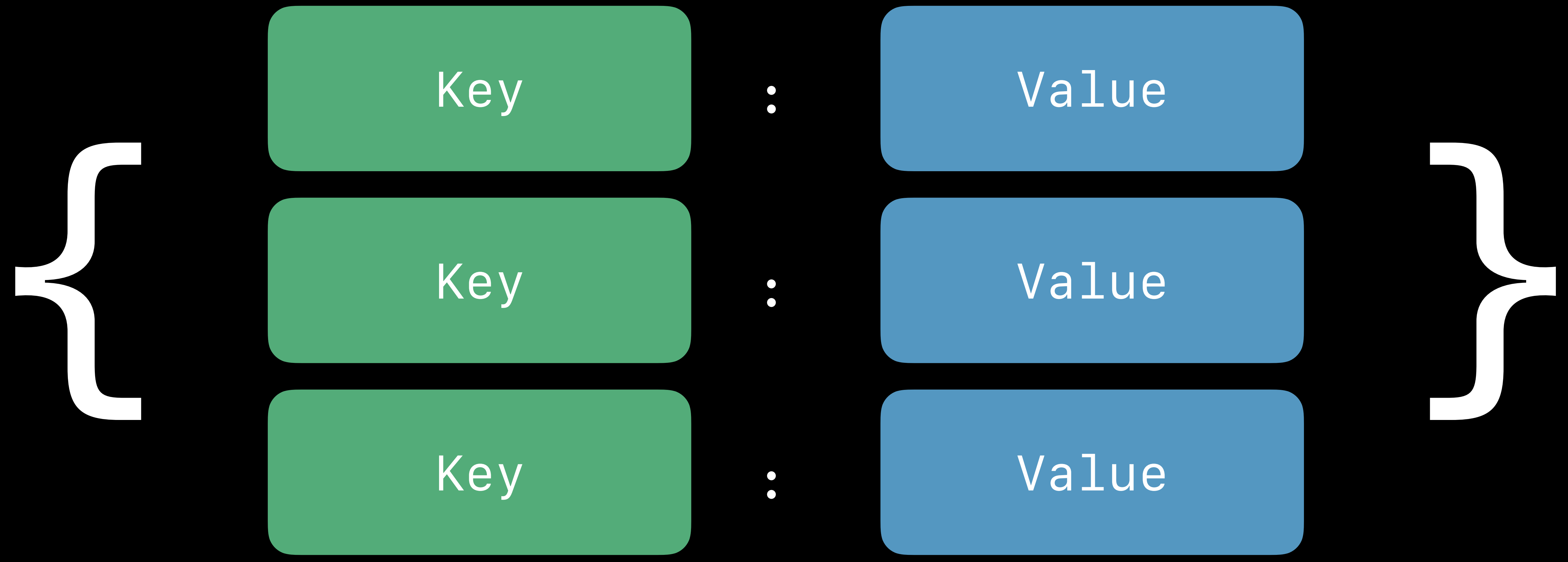
Encapsulate Encoding Details

Keys and values are private

Containers provide storage for values

Keyed Containers

Keyed Containers



Coding Keys

Strongly-typed replacement for String keys

```
public protocol CodingKey {
  var stringValue: String { get }
  var intValue: Int? { get }

  init?(stringValue: String)
  init?(intValue: Int)
}
```

Coding Keys

Coding Keys

```swift
private enum CodingKeys : String, CodingKey {
  case url
  case author
  case comment_count
}
```

Coding Keys

```
private enum CodingKeys : String, CodingKey {
  case url
  case author
  case comment_count
}
```

Case Name	stringValue	intValue?
url	url	nil
author	author	nil
comment_count	comment_count	nil

Coding Keys

```swift
private enum CodingKeys : String, CodingKey {
  case url
  case author
  case commentCount = "comment_count"
}
```

Case Name	stringValue	intValue?
url	url	nil
author	author	nil
commentCount	comment_count	nil

Coding Keys

```
private enum CodingKeys : Int, CodingKey {
  case url = 42
  case author = 100
  case comment_count
}
```

Case Name	stringValue	intValue?
url	url	42
author	author	100
comment_count	comment_count	101

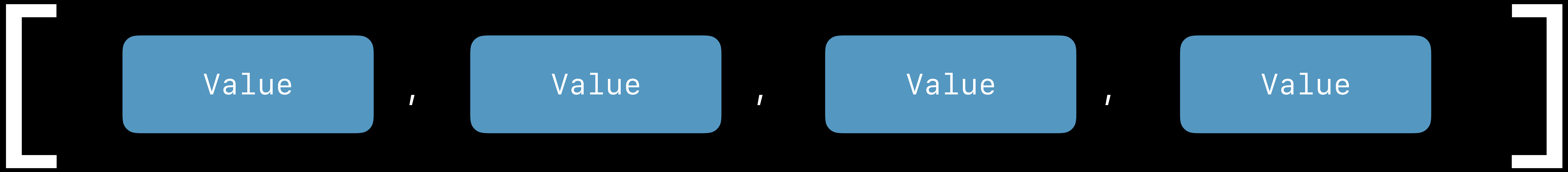
Coding Keys

```swift
private enum CodingKeys : Int, CodingKey {
    case url = 42
    case author = 100
    case comment_count
}
```

Case Name	stringValue	intValue?
url	url	42
author	author	100
comment_count	comment_count	101

Unkeyed Containers

Unkeyed Containers



Single Value Containers

Single Value Containers

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws { /* … */ }
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws { /* … */ }
  public func encode(to encoder: Encoder) throws {
    var container = encoder.container(keyedBy: CodingKeys.self)
    try container.encode(url, forKey: .url)
    try container.encode(message, forKey: .message)
    try container.encode(author, forKey: .author)
    try container.encode(commentCount, forKey: .commentCount)
  }
}
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws { /* … */ }
  public func encode(to encoder: Encoder) throws {
    var container = encoder.container(keyedBy: CodingKeys.self)
    try container.encode(url, forKey: .url)
    try container.encode(message, forKey: .message)
    try container.encode(author, forKey: .author)
    try container.encode(commentCount, forKey: .commentCount)
  }
}
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws { /* … */ }
  public func encode(to encoder: Encoder) throws {
    var container = encoder.container(keyedBy: CodingKeys.self)
    try container.encode(url, forKey: .url)
    try container.encode(message, forKey: .message)
    try container.encode(author, forKey: .author)
    try container.encode(commentCount, forKey: .commentCount)
  }
}
```

```swift
struct Commit : Codable {
  struct Author : Codable { /* … */ }
  let url: URL
  let message: String
  let author: Author
  let commentCount: Int
  private enum CodingKeys : String, CodingKey { /* … */ }
  public init(from decoder: Decoder) throws { /* … */ }
  public func encode(to encoder: Encoder) throws {
    var container = encoder.container(keyedBy: CodingKeys.self)
    try container.encode(url, forKey: .url)
    try container.encode(message, forKey: .message)
    try container.encode(author, forKey: .author)
    try container.encode(commentCount, forKey: .commentCount)
  }
}
```

```swift
struct Point2D : Encodable {
  var x: Double
  var y: Double
```

```swift
struct Point2D : Encodable {
  var x: Double
  var y: Double
```

```swift
struct Point2D : Encodable {
  var x: Double
  var y: Double

  public func encode(to encoder: Encoder) throws {
    var container = encoder.unkeyedContainer()
    try container.encode(x)
    try container.encode(y)
  }
}
```

```swift
struct Point2D : Encodable {
  var x: Double
  var y: Double

  public func encode(to encoder: Encoder) throws {
    var container = encoder.unkeyedContainer()
    try container.encode(x)
    try container.encode(y)
  }
}
```

```swift
struct Point2D : Encodable {
  var x: Double
  var y: Double

  public func encode(to encoder: Encoder) throws {
    var container = encoder.unkeyedContainer()
    try container.encode(x)
    try container.encode(y)
  }
}

// [ 1.5, 3.9 ]
```

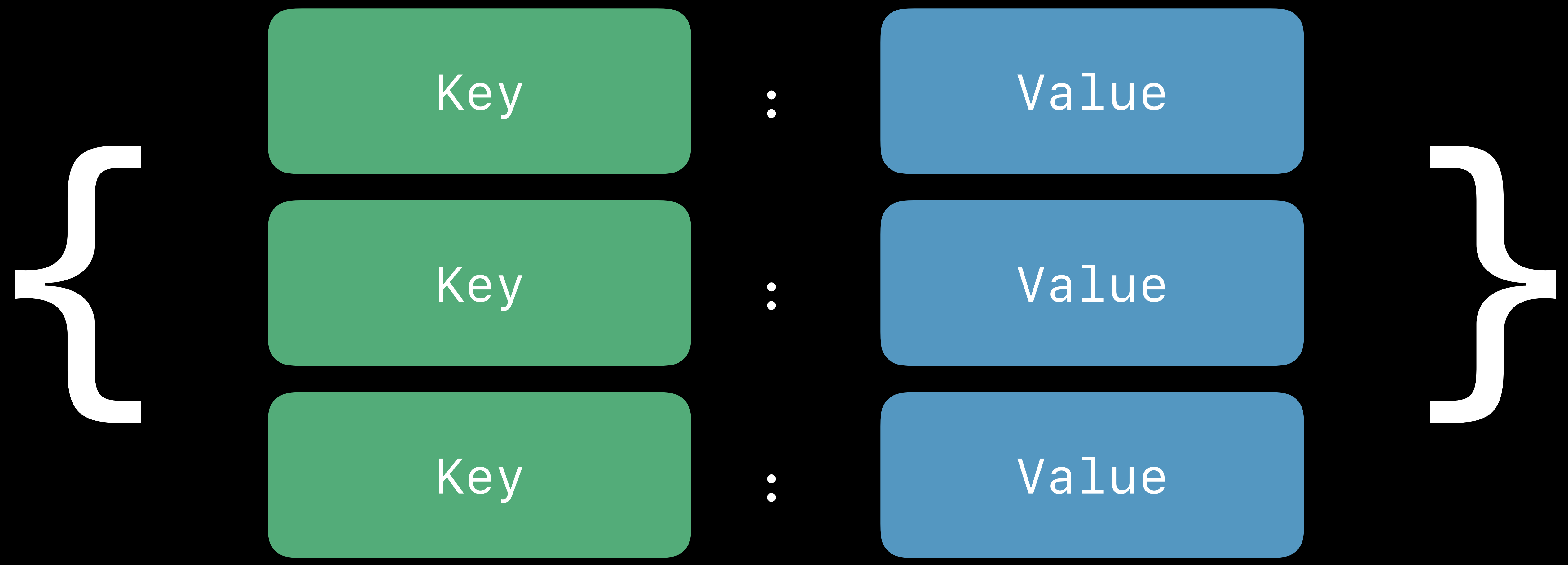
```swift
struct Point2D : Encodable {
  var x: Double
  var y: Double

  public func encode(to encoder: Encoder) throws {
    var container = encoder.unkeyedContainer()
    try container.encode(x)
    try container.encode(y)
  }
}
```

```
// [ 1.5, 3.9 ]
```

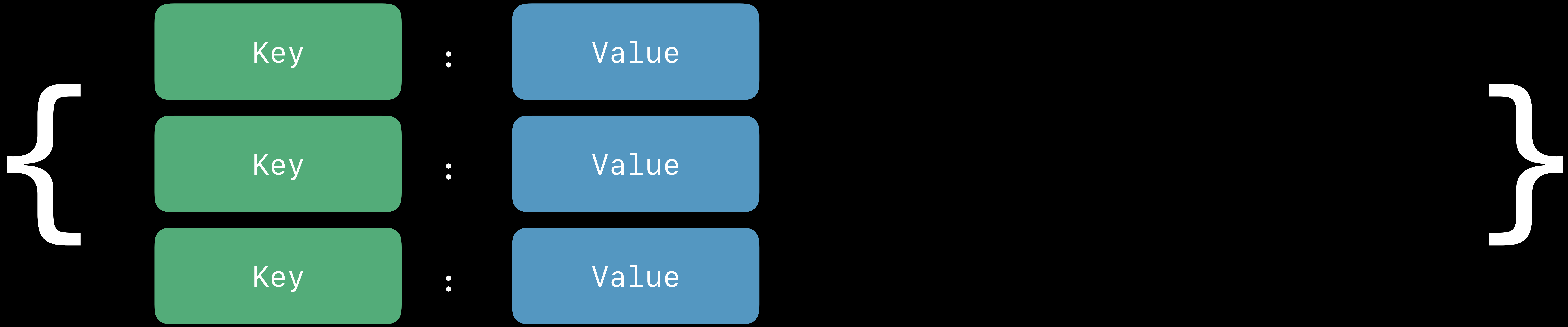
Nested Containers

Lightweight encapsulation of additional values



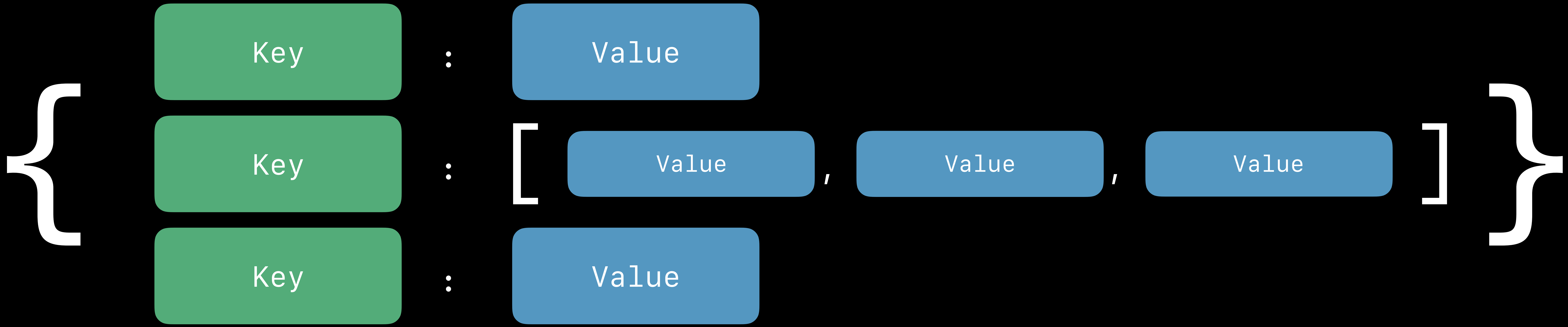
Nested Containers

Lightweight encapsulation of additional values



Nested Containers

Lightweight encapsulation of additional values



Encoding a Class Hierarchy

Use nested container for superclass data

Encapsulates keys and values from superclass

```swift
class Animal : Decodable {
  var legCount: Int
  private enum CodingKeys: String, CodingKey { case legCount }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    legCount = try container.decode(Int.self, forKey: .legCount)
  }
}
```

```swift
class Animal : Decodable {
  var legCount: Int
  private enum CodingKeys: String, CodingKey { case legCount }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    legCount = try container.decode(Int.self, forKey: .legCount)
  }
}
```

```swift
class Animal : Decodable {
  var legCount: Int
  private enum CodingKeys: String, CodingKey { case legCount }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    legCount = try container.decode(Int.self, forKey: .legCount)
  }
}
class Dog : Animal {
  var bestFriend: Kid
  private enum CodingKeys : String, CodingKey { case bestFriend }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    bestFriend = try container.decode(Kid.self, forKey: .bestFriend)
    let superDecoder = try container.superDecoder()
    try super.init(from: superDecoder)
  }
}
```

```swift
class Animal : Decodable {
  var legCount: Int
  private enum CodingKeys: String, CodingKey { case legCount }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    legCount = try container.decode(Int.self, forKey: .legCount)
  }
}
class Dog : Animal {
  var bestFriend: Kid
  private enum CodingKeys : String, CodingKey { case bestFriend }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    bestFriend = try container.decode(Kid.self, forKey: .bestFriend)
    let superDecoder = try container.superDecoder()
    try super.init(from: superDecoder)
  }
```

```swift
class Animal : Decodable {
  var legCount: Int
  private enum CodingKeys: String, CodingKey { case legCount }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    legCount = try container.decode(Int.self, forKey: .legCount)
  }
}
class Dog : Animal {
  var bestFriend: Kid
  private enum CodingKeys : String, CodingKey { case bestFriend }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    bestFriend = try container.decode(Kid.self, forKey: .bestFriend)
    let superDecoder = try container.superDecoder()
    try super.init(from: superDecoder)
  }
```

```swift
class Animal : Decodable {
  var legCount: Int
  private enum CodingKeys: String, CodingKey { case legCount }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    legCount = try container.decode(Int.self, forKey: .legCount)
  }
}
class Dog : Animal {
  var bestFriend: Kid
  private enum CodingKeys : String, CodingKey { case bestFriend }
  required init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    bestFriend = try container.decode(Kid.self, forKey: .bestFriend)
    let superDecoder = try container.superDecoder()
    try super.init(from: superDecoder)
  }
}
```

Codable Philosophy

Error handling built-in

Encapsulate encoding details

Abstract format from types

Abstract Format from Types

Reuse one implementation of `Encodable` and `Decodable`

Abstract Format from Types

Reuse one implementation of `Encodable` and `Decodable`

Allow new formats without library changes

Abstract Format from Types

Reuse one implementation of `Encodable` and `Decodable`

Allow new formats without library changes

Formats have different fundamental types and conventions

Encoding Strategies

Encoder-specific customizations for certain types

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

```
"2017-06-07T18:00:40Z"
```

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

```
1496858440.0729699
```

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

```
1496858440072.97
```

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

```
"Wednesday, June 7, 2017 at 11:00 AM"
```

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

```
"Wednesday, June 7, 2017 at 11:00 AM"
```

`Data`

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

```
"Wednesday, June 7, 2017 at 11:00 AM"
```

`Data`

```
"AAIABAA="
```

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

```
"Wednesday, June 7, 2017 at 11:00 AM"
```

`Data`

```
[0,2,0,4,0]
```

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

```
"Wednesday, June 7, 2017 at 11:00 AM"
```

`Data`

```
"🐏🐶🐏🐶🐏"
```

Encoding Strategies

Encoder-specific customizations for certain types

JSON

`Date`

```
"Wednesday, June 7, 2017 at 11:00 AM"
```

`Data`

```
"🐏🐶🐏🐶🐏"
```

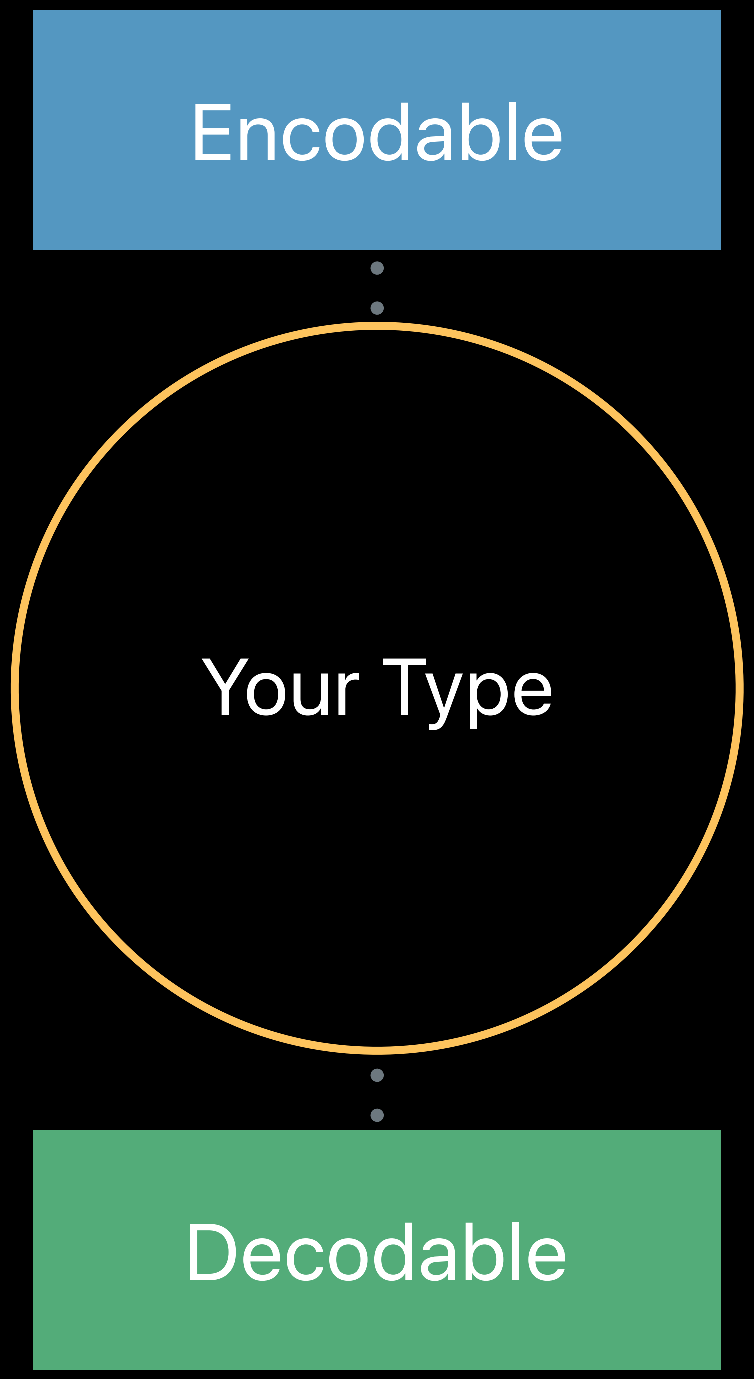
Property Lists

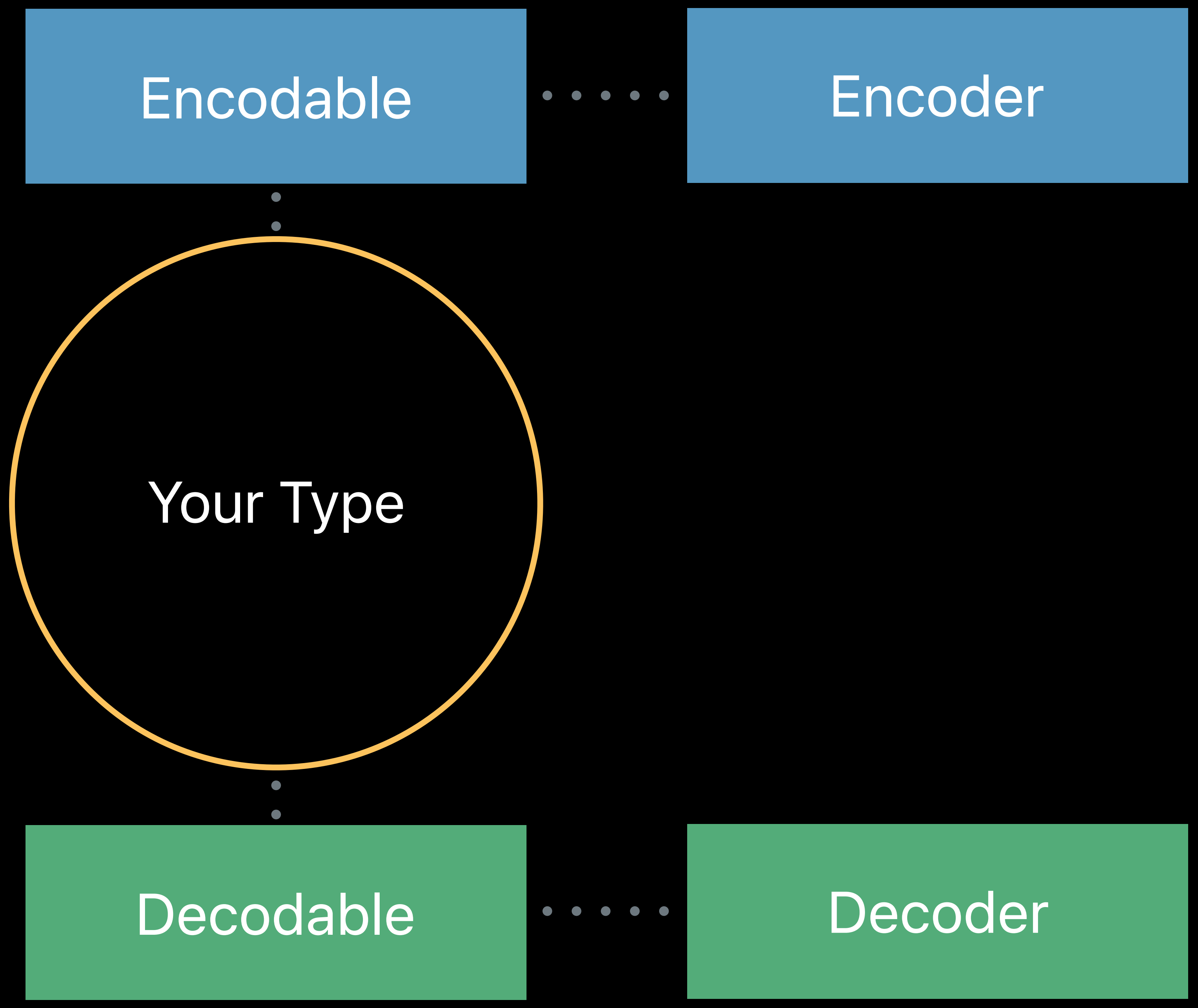
Codable Foundation Types

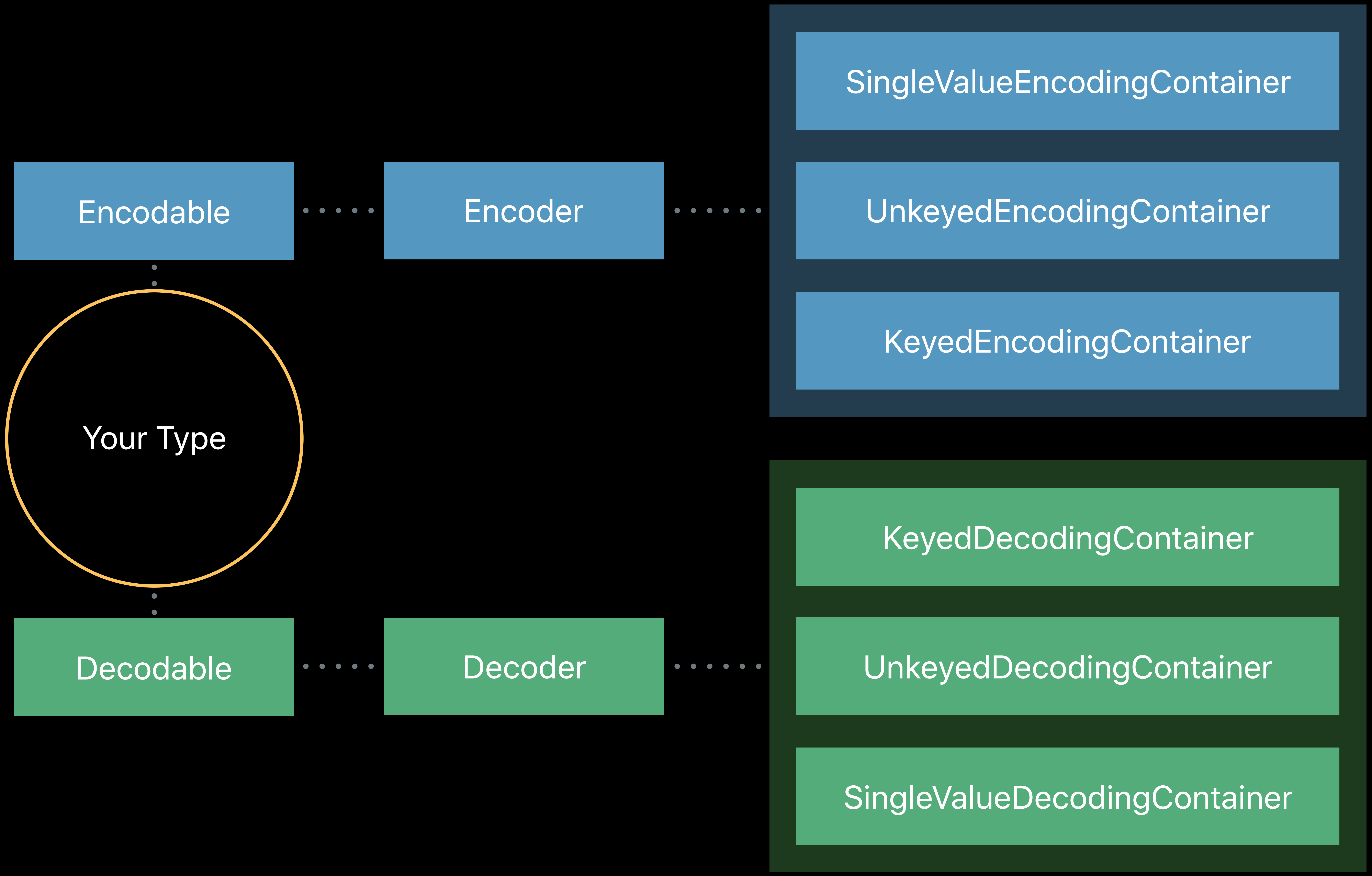
`CGFloat`	`IndexPath`
`AffineTransform`	`IndexSet`
`Calendar`	`Locale`
`CharacterSet`	`Measurement`
`Data`	`NSRange`
`Date`	`PersonNameComponents`
`DateComponents`	`TimeZone`
`DateInterval`	`URL`
`Decimal`	`UUID`

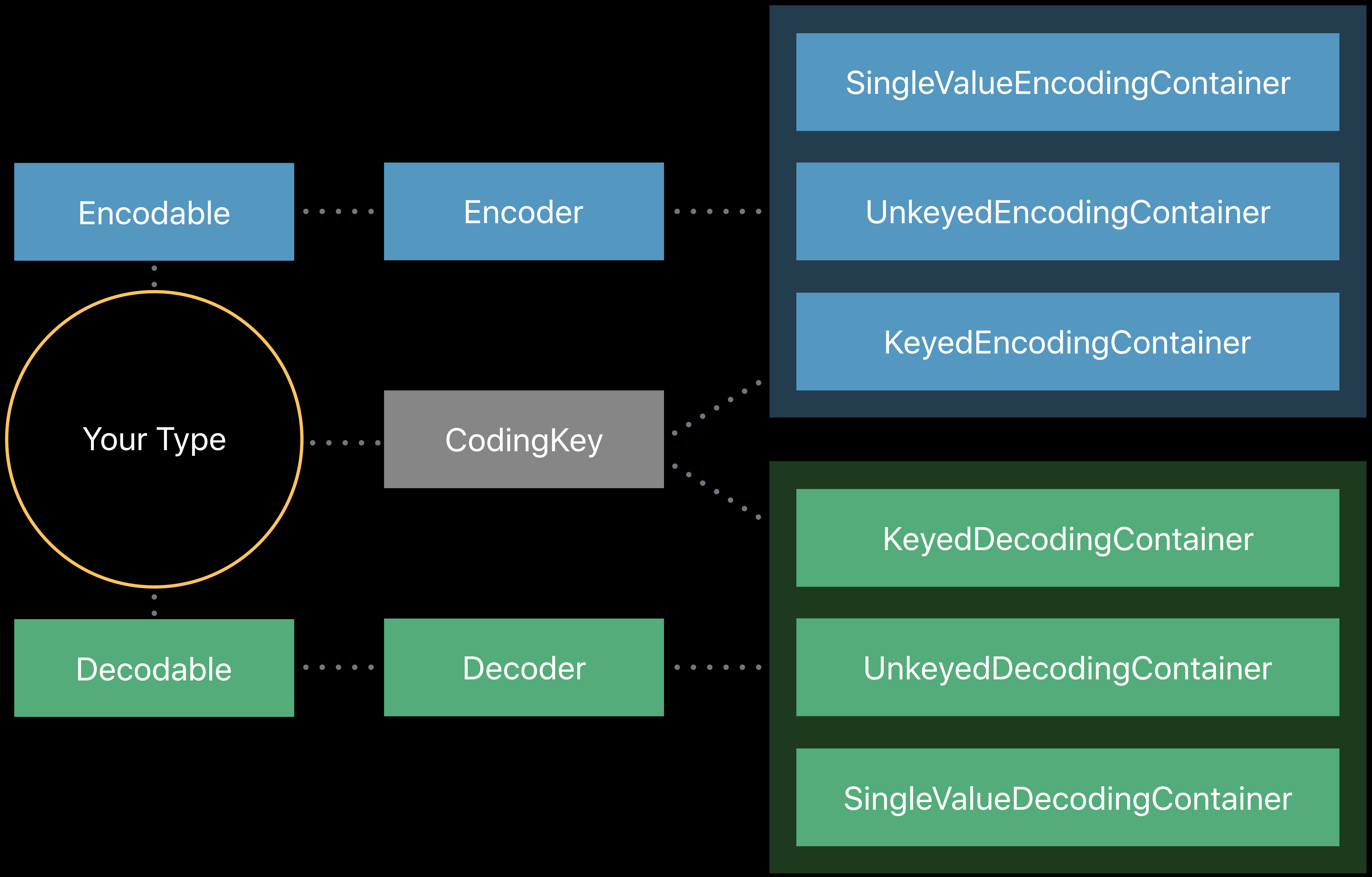
Your Type

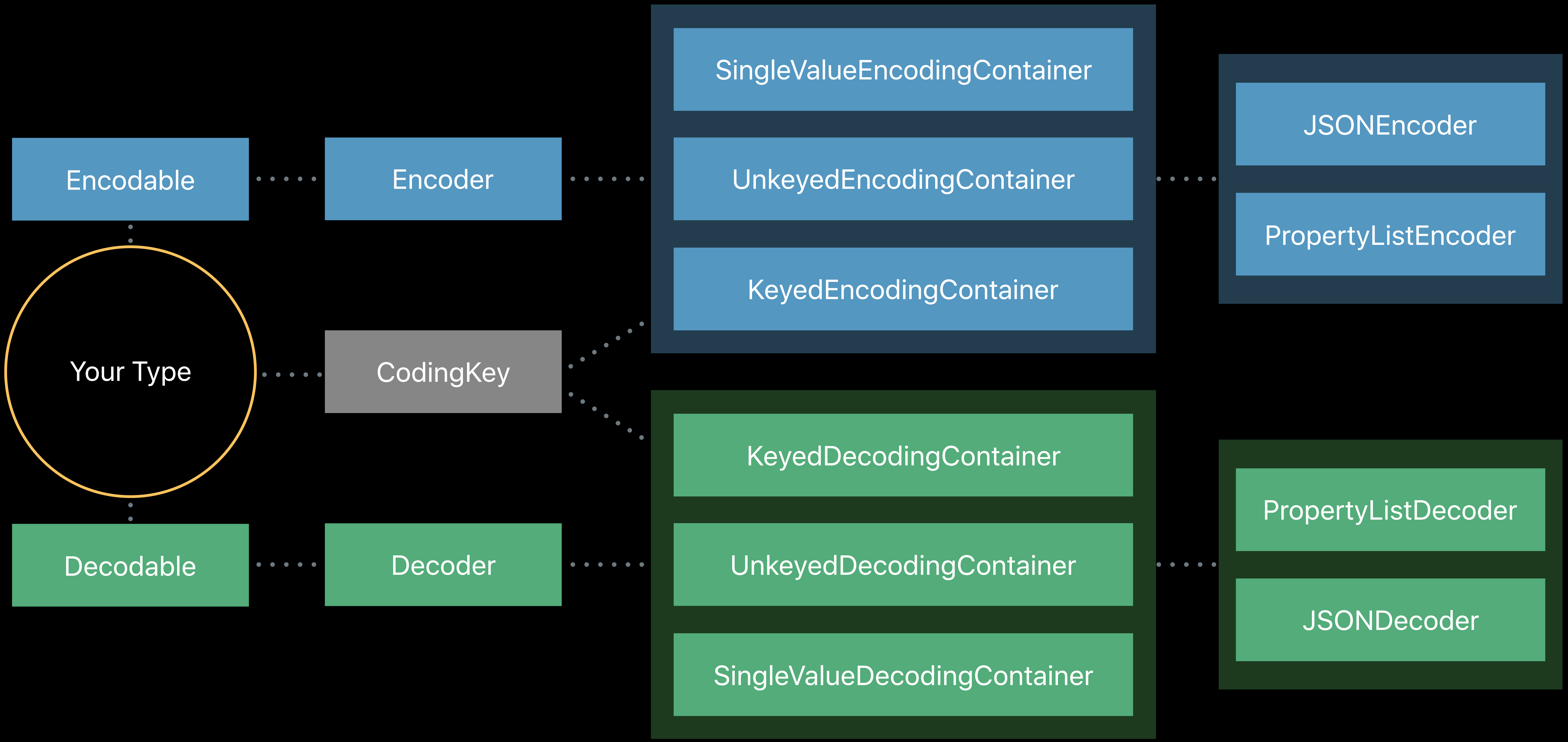
Your Type


Encodable
Your Type
Decodable


Encodable
Encoder
Your Type
Decodable
Decoder


SingleValueEncodingContainer
UnkeyedEncodingContainer
KeyedEncodingContainer
Encodable
Encoder
Your Type
Decodable
Decoder
KeyedDecodingContainer
UnkeyedDecodingContainer
SingleValueDecodingContainer


Encodable
Encoder
SingleValueEncodingContainer
UnkeyedEncodingContainer
KeyedEncodingContainer
Your Type
CodingKey
KeyedDecodingContainer
UnkeyedDecodingContainer
SingleValueDecodingContainer
Decodable
Decoder


Encodable
Encoder
SingleValueEncodingContainer
UnkeyedEncodingContainer
KeyedEncodingContainer
JSONEncoder
PropertyListEncoder
Your Type
CodingKey
KeyedDecodingContainer
UnkeyedDecodingContainer
SingleValueDecodingContainer
PropertyListDecoder
JSONDecoder
Decodable
Decoder

Summary

New API and improved performance in Foundation

Strongly typed key paths for Swift

New Key-Value Observation API

New `Codable` protocols