peterfriese / Color+Codable.swift

Created 2 years ago • Report abuse

Making Swift's Color codable

Color+Codable.swift

```swift
//
//  Color+Codable.swift
//  FirestoreCodableSamples
//
//  Created by Peter Friese on 18.03.21.
//

import SwiftUI

// Inspired by https://cocoacasts.com/from-hex-to-uicolor-and-back-in-swift
// Make Color codable. This includes support for transparency.
// See https://www.digitalocean.com/community/tutorials/css-hex-code-colors-alpha-values
extension Color: Codable {
  init(hex: String) {
    let rgba = hex.toRGBA()

    self.init(.sRGB,
              red: Double(rgba.r),
              green: Double(rgba.g),
              blue: Double(rgba.b),
              opacity: Double(rgba.alpha))
  }

  public init(from decoder: Decoder) throws {
    let container = try decoder.singleValueContainer()
    let hex = try container.decode(String.self)

    self.init(hex: hex)
  }

  public func encode(to encoder: Encoder) throws {
    var container = encoder.singleValueContainer()
    try container.encode(toHex)
  }

```

```
  var toHex: String? {
    return toHex()
  }

  func toHex(alpha: Bool = false) -> String? {
    guard let components = cgColor?.components, components.count >= 3 else {
      return nil
    }

    let r = Float(components[0])
    let g = Float(components[1])
    let b = Float(components[2])
    var a = Float(1.0)

    if components.count >= 4 {
      a = Float(components[3])
    }

    if alpha {
      return String(format: "%02lX%02lX%02lX%02lX",
                    lroundf(r * 255),
                    lroundf(g * 255),
                    lroundf(b * 255),
                    lroundf(a * 255))
    }
    else {
      return String(format: "%02lX%02lX%02lX",
                    lroundf(r * 255),
                    lroundf(g * 255),
                    lroundf(b * 255))
    }
  }
}

extension String {
  func toRGBA() -> (r: CGFloat, g: CGFloat, b: CGFloat, alpha: CGFloat) {
    var hexSanitized = self.trimmingCharacters(in: .whitespacesAndNewlines)
    hexSanitized = hexSanitized.replacingOccurrences(of: "#", with: "")

    var rgb: UInt64 = 0

    var r: CGFloat = 0.0
    var g: CGFloat = 0.0
    var b: CGFloat = 0.0
    var a: CGFloat = 1.0

    let length = hexSanitized.count

    Scanner(string: hexSanitized).scanHexInt64(&rgb)
```

```
85
86      if length == 6 {
87        r = CGFloat((rgb & 0xFF0000) >> 16) / 255.0
88        g = CGFloat((rgb & 0x00FF00) >> 8) / 255.0
89        b = CGFloat(rgb & 0x0000FF) / 255.0
90      }
91      else if length == 8 {
92        r = CGFloat((rgb & 0xFF000000) >> 24) / 255.0
93        g = CGFloat((rgb & 0x00FF0000) >> 16) / 255.0
94        b = CGFloat((rgb & 0x0000FF00) >> 8) / 255.0
95        a = CGFloat(rgb & 0x000000FF) / 255.0
96      }
97
98      return (r, g, b, a)
99    }
100 }
```