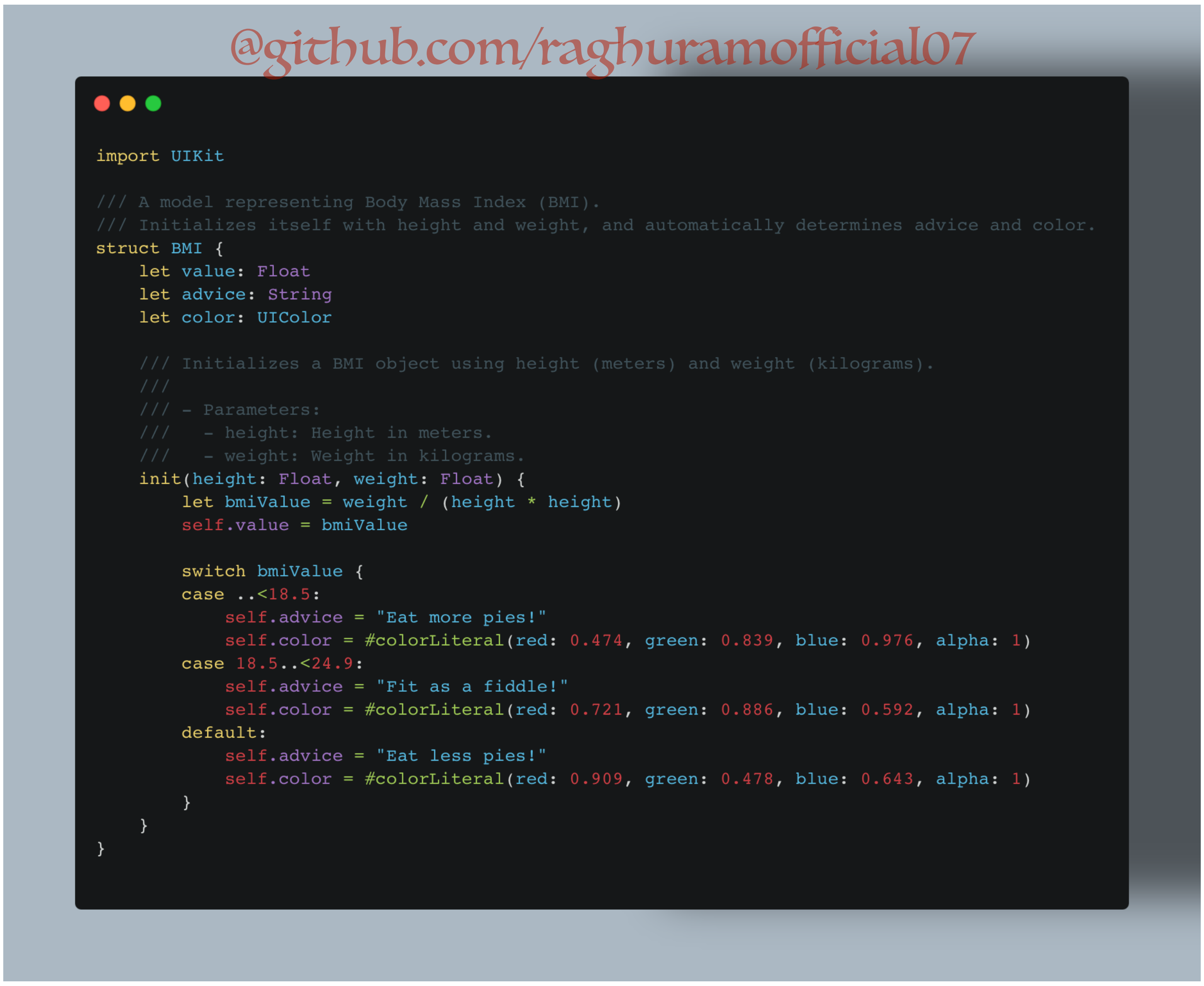
Body Mass Index (BMI)

Documentation

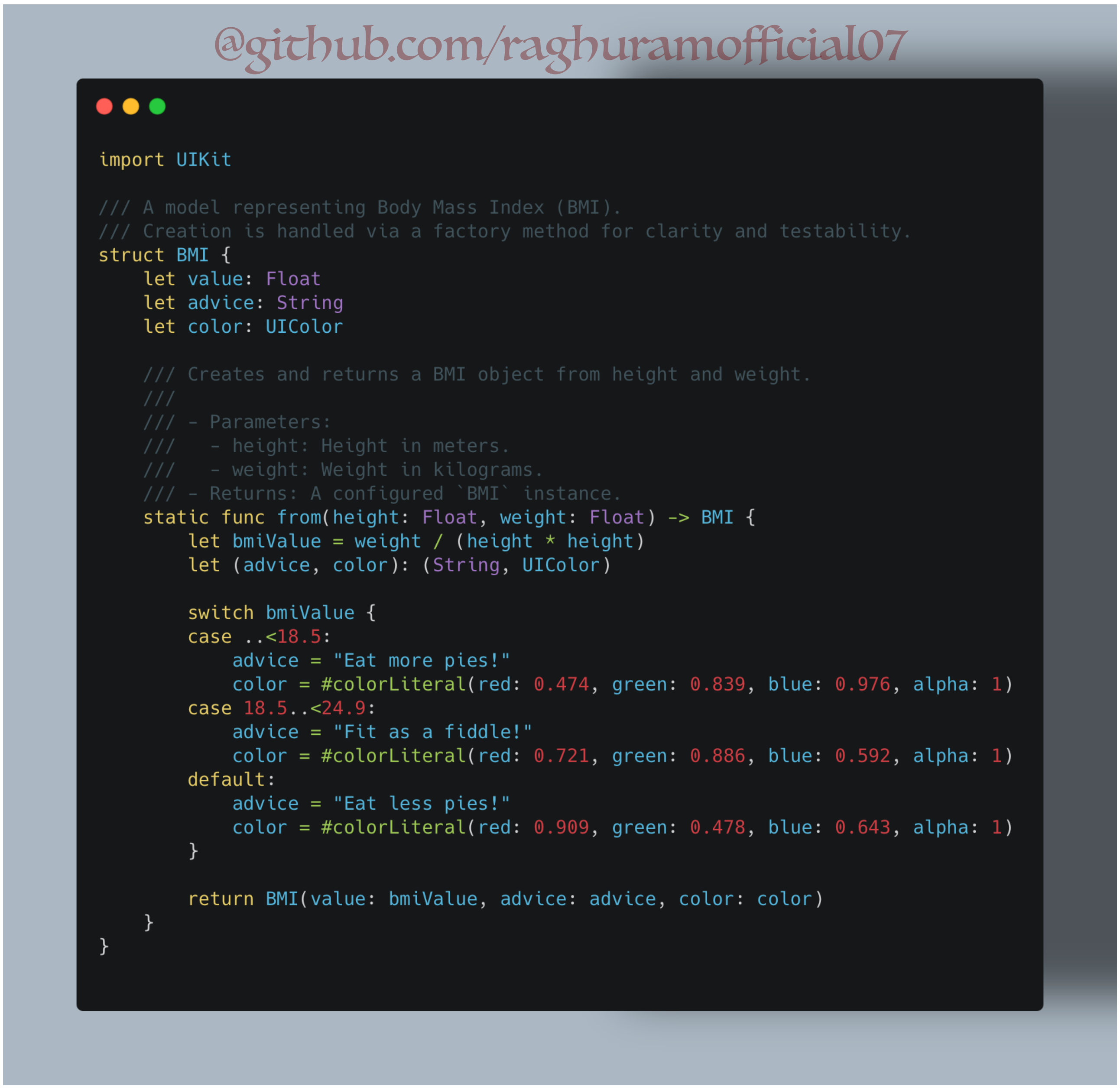
@github.com/raghuramofficial07

Approach (1)



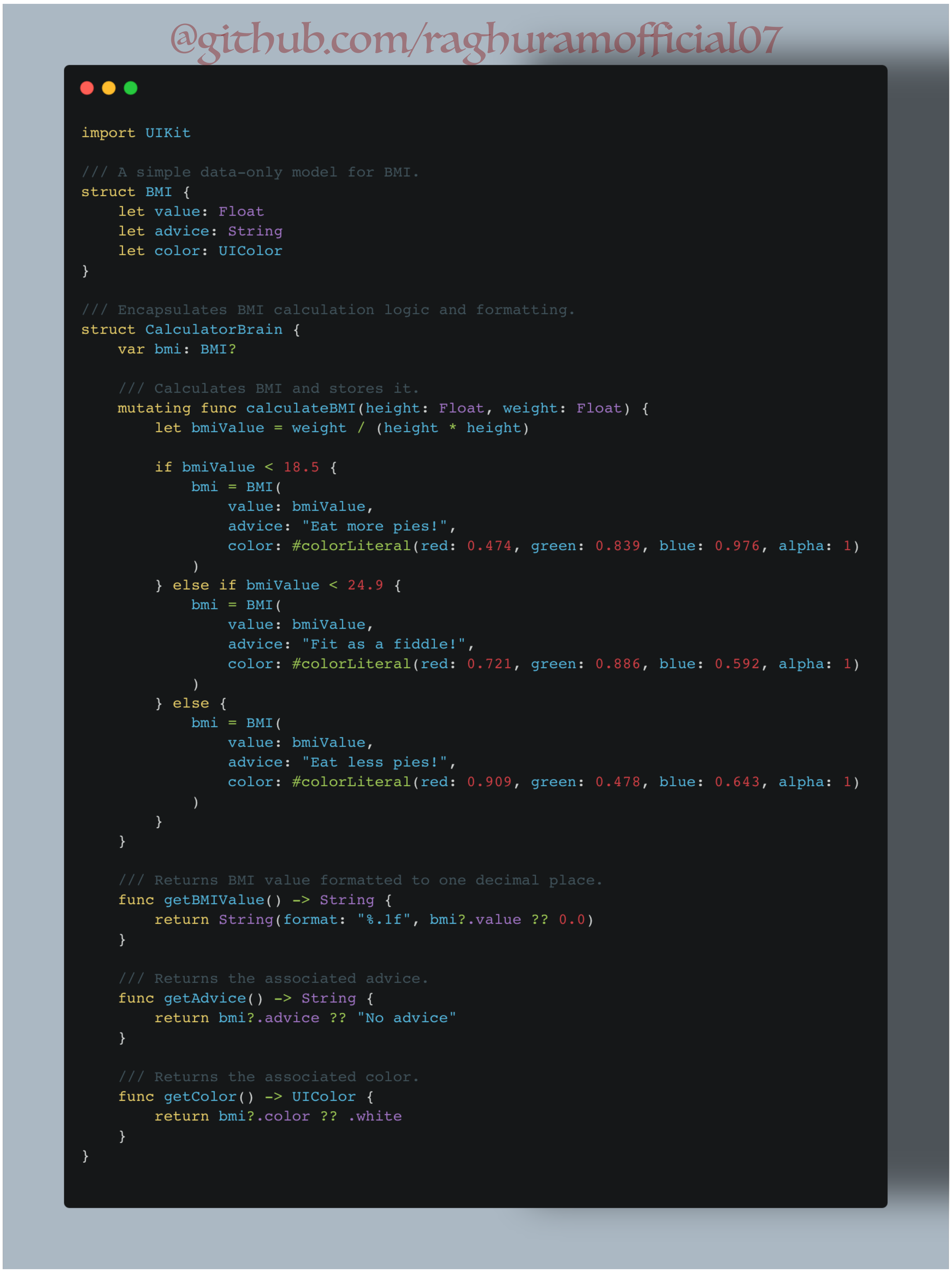
```swift
import UIKit

/// A model representing Body Mass Index (BMI).
/// Initializes itself with height and weight, and automatically determines advice and color.
struct BMI {
    let value: Float
    let advice: String
    let color: UIColor

    /// Initializes a BMI object using height (meters) and weight (kilograms).
    ///
    /// - Parameters:
    ///   - height: Height in meters.
    ///   - weight: Weight in kilograms.
    init(height: Float, weight: Float) {
        let bmiValue = weight / (height * height)
        self.value = bmiValue

        switch bmiValue {
        case ..<18.5:
            self.advice = "Eat more pies!"
            self.color = #colorLiteral(red: 0.474, green: 0.839, blue: 0.976, alpha: 1)
        case 18.5..<24.9:
            self.advice = "Fit as a fiddle!"
            self.color = #colorLiteral(red: 0.721, green: 0.886, blue: 0.592, alpha: 1)
        default:
            self.advice = "Eat less pies!"
            self.color = #colorLiteral(red: 0.909, green: 0.478, blue: 0.643, alpha: 1)
        }
    }
}
```

Approach (2)



```swift
import UIKit

/// A model representing Body Mass Index (BMI).
/// Creation is handled via a factory method for clarity and testability.
struct BMI {
    let value: Float
    let advice: String
    let color: UIColor

    /// Creates and returns a BMI object from height and weight.
    ///
    /// - Parameters:
    ///   - height: Height in meters.
    ///   - weight: Weight in kilograms.
    /// - Returns: A configured `BMI` instance.
    static func from(height: Float, weight: Float) -> BMI {
        let bmiValue = weight / (height * height)
        let (advice, color): (String, UIColor)

        switch bmiValue {
        case ..<18.5:
            advice = "Eat more pies!"
            color = #colorLiteral(red: 0.474, green: 0.839, blue: 0.976, alpha: 1)
        case 18.5..<24.9:
            advice = "Fit as a fiddle!"
            color = #colorLiteral(red: 0.721, green: 0.886, blue: 0.592, alpha: 1)
        default:
            advice = "Eat less pies!"
            color = #colorLiteral(red: 0.909, green: 0.478, blue: 0.643, alpha: 1)
        }

        return BMI(value: bmiValue, advice: advice, color: color)
    }
}
```

Approach (3)



```swift
import UIKit

/// A simple data-only model for BMI.
struct BMI {
    let value: Float
    let advice: String
    let color: UIColor
}

/// Encapsulates BMI calculation logic and formatting.
struct CalculatorBrain {
    var bmi: BMI?

    /// Calculates BMI and stores it.
    mutating func calculateBMI(height: Float, weight: Float) {
        let bmiValue = weight / (height * height)

        if bmiValue < 18.5 {
            bmi = BMI(
                value: bmiValue,
                advice: "Eat more pies!",
                color: #colorLiteral(red: 0.474, green: 0.839, blue: 0.976, alpha: 1)
            )
        } else if bmiValue < 24.9 {
            bmi = BMI(
                value: bmiValue,
                advice: "Fit as a fiddle!",
                color: #colorLiteral(red: 0.721, green: 0.886, blue: 0.592, alpha: 1)
            )
        } else {
            bmi = BMI(
                value: bmiValue,
                advice: "Eat less pies!",
                color: #colorLiteral(red: 0.909, green: 0.478, blue: 0.643, alpha: 1)
            )
        }
    }

    /// Returns BMI value formatted to one decimal place.
    func getBMIValue() -> String {
        return String(format: "%.1f", bmi?.value ?? 0.0)
    }

    /// Returns the associated advice.
    func getAdvice() -> String {
        return bmi?.advice ?? "No advice"
    }

    /// Returns the associated color.
    func getColor() -> UIColor {
        return bmi?.color ?? .white
    }
}
```

Description :

🧠 Understanding `var bmi: BMI?`

In Swift, the declaration:

```
var bmi: BMI?
```

defines a variable named `bmi` that can hold an instance of the `BMI` struct or be `nil` (representing the absence of a value). This is known as an **optional** type, denoted by the `?` symbol.

🔍 What Does This Mean?

- **Optional Type**: The `?` indicates that the variable can either contain a value of type `BMI` or be `nil`. This is a feature in Swift that allows variables to have a "no-value" state, which is useful for representing situations where a value is not yet set or is unavailable.
- **Mutable Variable**: The `var` keyword signifies that `bmi` is a variable, meaning its value can be changed after it's initially set. This is important in scenarios where the BMI value might be recalculated or updated based on new data.

🧩 Why Use an Optional?

Using an optional for `bmi` serves several purposes:

1. **Represents Absence of Value**: Before the user inputs their height and weight, the BMI hasn't been calculated yet. Declaring `bmi` as an optional allows it to be `nil` initially, clearly indicating the absence of a value.
2. **Safe Handling of Missing Data**: Swift's optional types help prevent runtime errors by forcing developers to handle the case where a value might be `nil`. This leads to safer and more predictable code.
3. **Flexibility in Data Assignment**: Since `bmi` is a variable, it can be updated or reset as needed, allowing the application to reflect changes in the user's data or calculations.

📘 **Apple Documentation Reference : ClickHere**