Concurrency-Safe Testing in Swift 6.1 with @TaskLocal and Test Scoping

DON'T GET LEFT BEHIND!

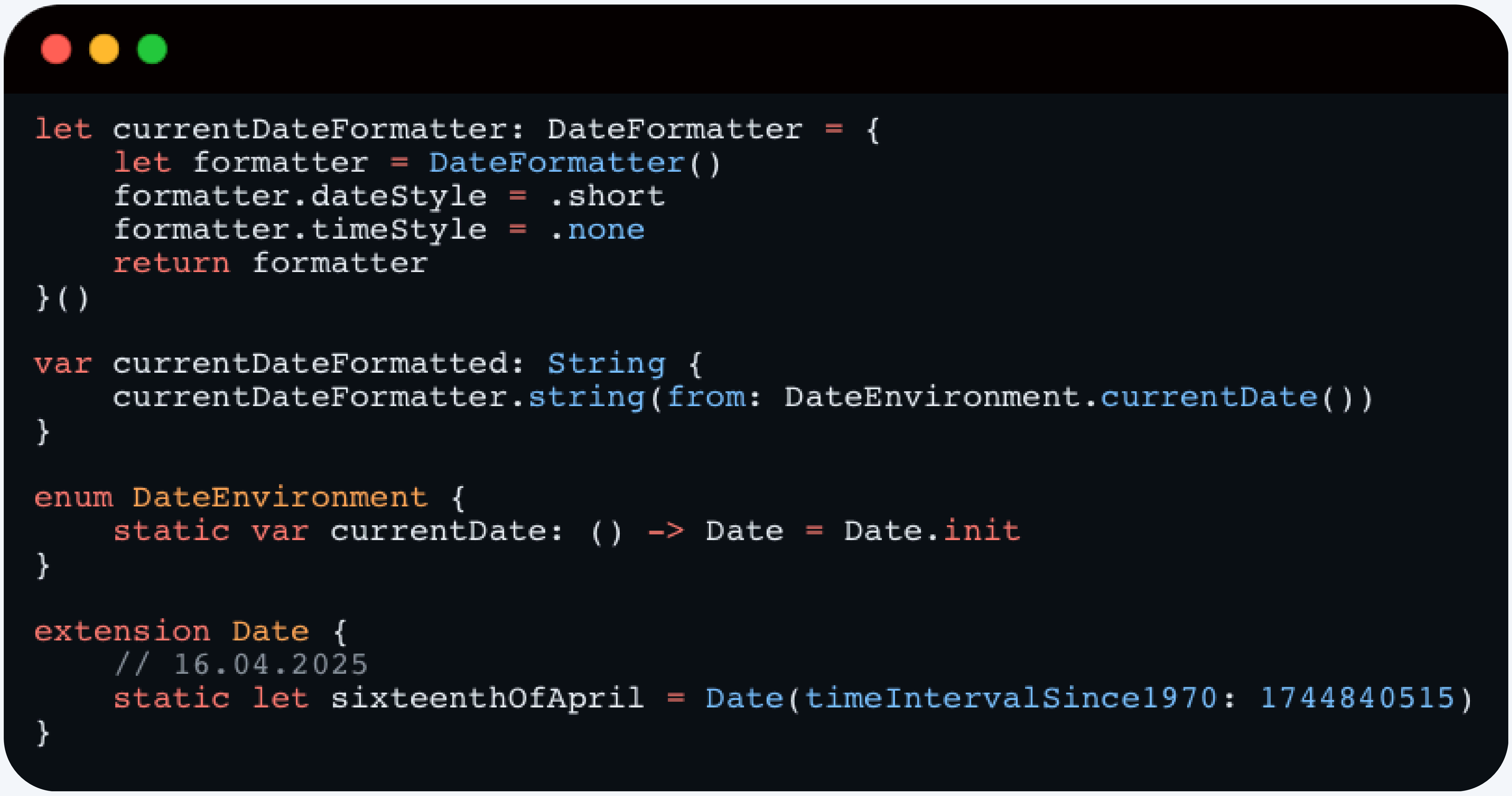
Maciej Gomółka

Initial setup

Today's example shows a static property providing the current date.

Perfect mechanism when we don't want to inject it everywhere where it's used.

How to test it? Scroll to find out!

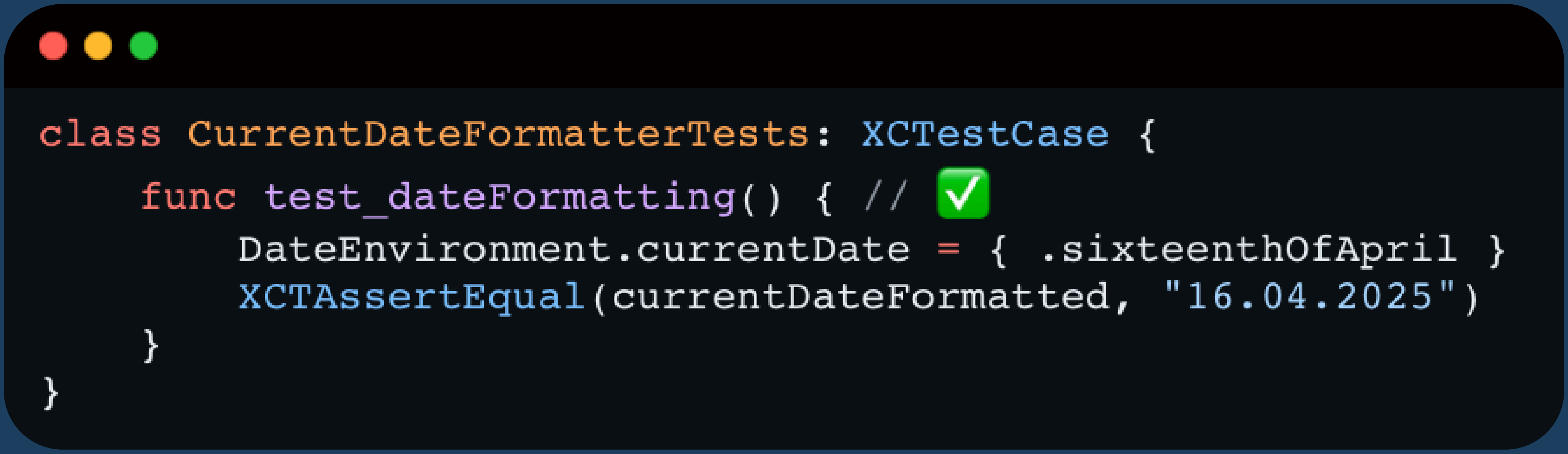


```swift
let currentDateFormatter: DateFormatter = {
    let formatter = DateFormatter()
    formatter.dateStyle = .short
    formatter.timeStyle = .none
    return formatter
}()

var currentDateFormatted: String {
    currentDateFormatter.string(from: DateEnvironment.currentDate())
}

enum DateEnvironment {
    static var currentDate: () -> Date = Date.init
}

extension Date {
    // 16.04.2025
    static let sixteenthOfApril = Date(timeIntervalSince1970: 1744840515)
}
```

Maciej Gomółka

Old way - XCTest

- Override the static property in tests.
- Works, but not concurrency-safe (Ok, for XCTest because test ran serially).



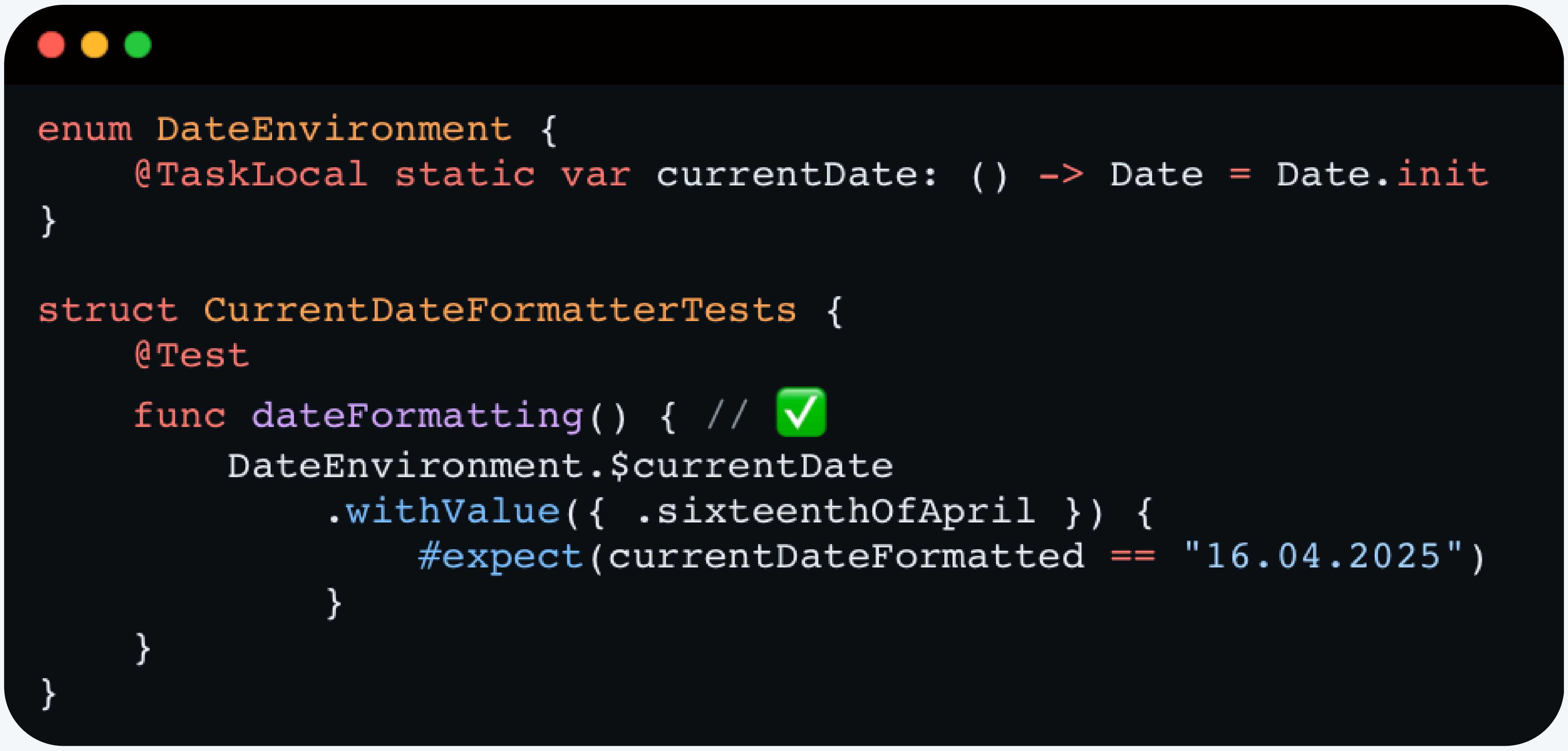
```swift
class CurrentDateFormatterTests: XCTestCase {
    func test_dateFormatting() { // ✅
        DateEnvironment.currentDate = { .sixteenthOfApril }
        XCTAssertEqual(currentDateFormatted, "16.04.2025")
    }
}
```

Maciej Gomółka

Swift Testing before Swift 6.1

Why @TaskLocal?

- Scoped per async Task.
- No global side-effects
- Safe for parallel tests

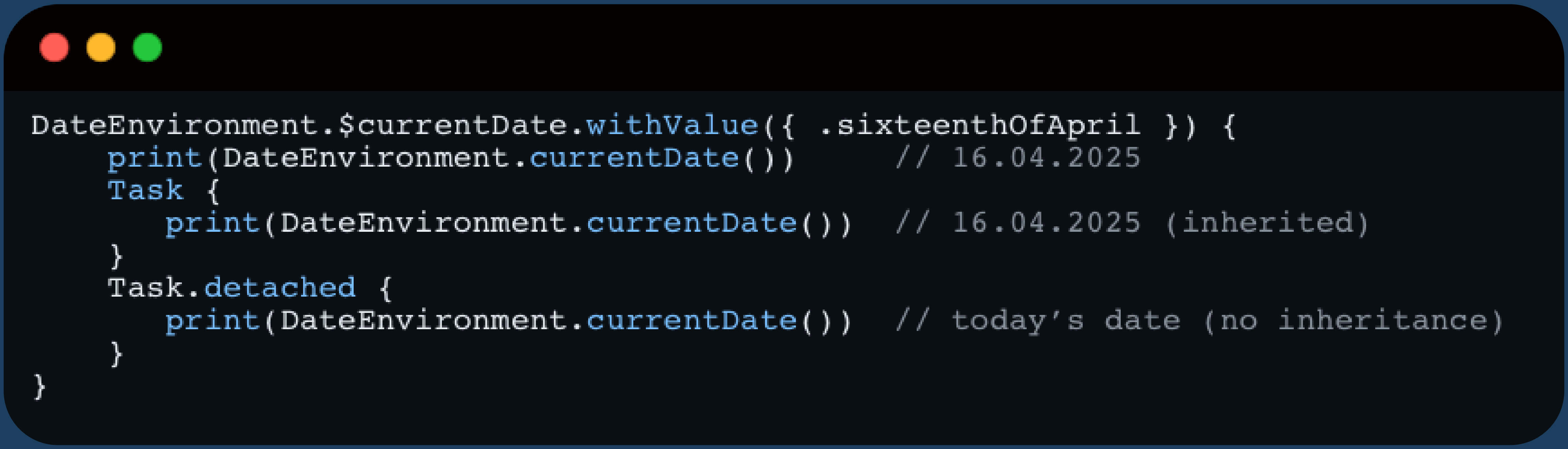


```swift
enum DateEnvironment {
    @TaskLocal static var currentDate: () -> Date = Date.init
}

struct CurrentDateFormatterTests {
    @Test
    func dateFormatting() { // ✅
        DateEnvironment.$currentDate
            .withValue({ .sixteenthOfApril }) {
                #expect(currentDateFormatted == "16.04.2025")
            }
    }
}
```

Maciej Gomółka

What is @TaskLocal?

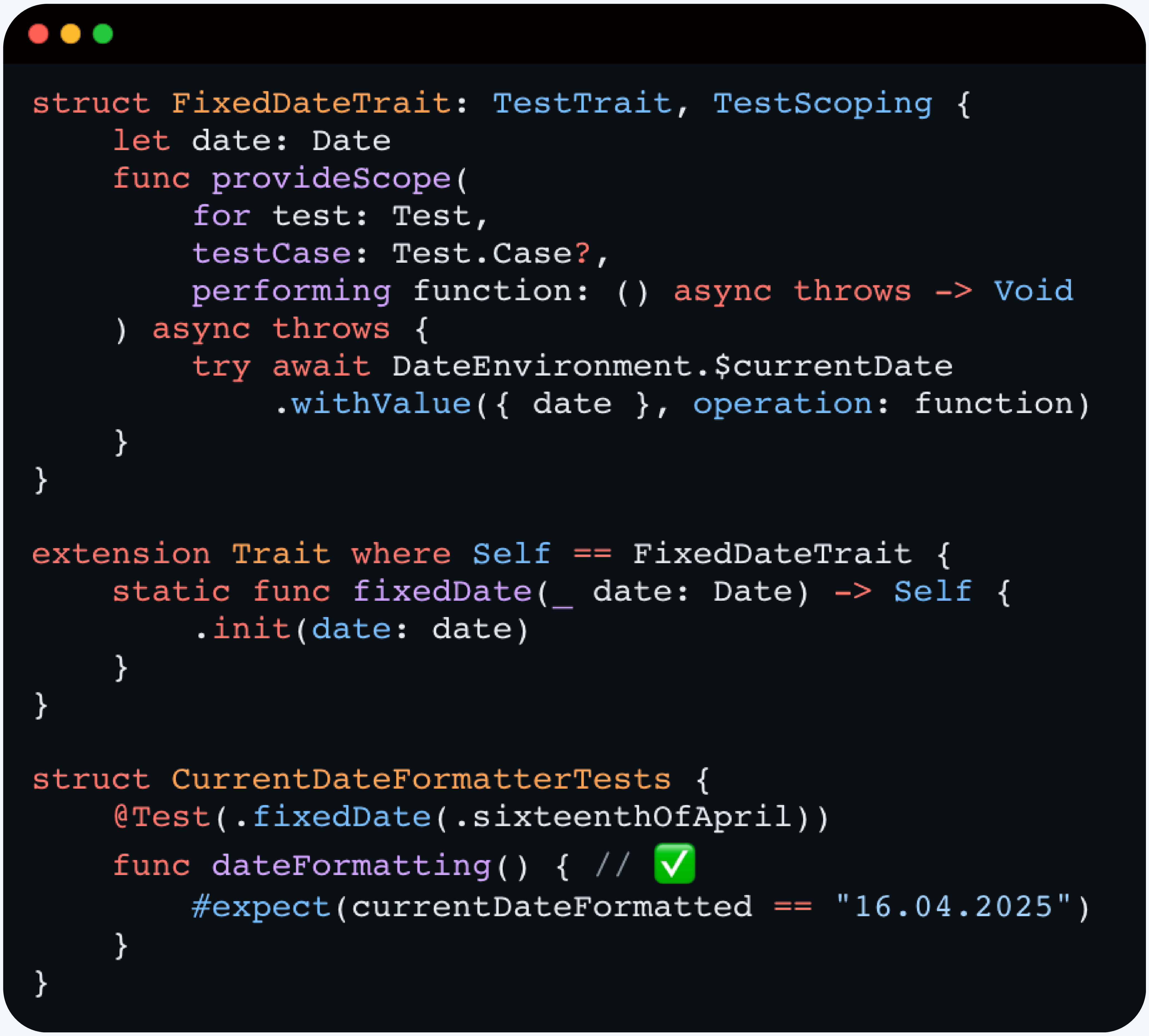
- Property wrapper for **per-Task** storage.
- **Default** value is returned when no override is applied.
- withValue(_:, operation:) - binds a new value **just** for that Task.
- **Child** Task { … } inherits override.
- **Detached** - Task.detached { … } does not inherit override.



```swift
DateEnvironment.$currentDate.withValue({ .sixteenthOfApril }) {
    print(DateEnvironment.currentDate())     // 16.04.2025
    Task {
       print(DateEnvironment.currentDate())  // 16.04.2025 (inherited)
    }
    Task.detached {
       print(DateEnvironment.currentDate())  // today's date (no inheritance)
    }
}
```

Maciej Gomółka

New in Swift 6.1: Test Scoping in Action



```swift
struct FixedDateTrait: TestTrait, TestScoping {
    let date: Date
    func provideScope(
        for test: Test,
        testCase: Test.Case?,
        performing function: () async throws -> Void
    ) async throws {
        try await DateEnvironment.$currentDate
            .withValue({ date }, operation: function)
    }
}

extension Trait where Self == FixedDateTrait {
    static func fixedDate(_ date: Date) -> Self {
        .init(date: date)
    }
}

struct CurrentDateFormatterTests {
    @Test(.fixedDate(.sixteenthOfApril))
    func dateFormatting() { // ✅
        #expect(currentDateFormatted == "16.04.2025")
    }
}
```

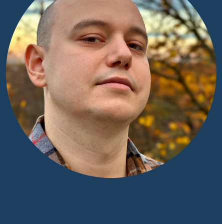
Maciej Gomółka

My final advice

Stay Curious

Use @TaskLocal and Test Scoping for deterministic, parallel-safe tests. Enjoy faster feedback looks and cleaner code.

Remember - Learning never stops - keep experimenting with new features!



Maciej Gomółka

Let's Connect!

🔷 **Follow** for more Swift & testing tips 🔷
🔷 **Comment** your thoughts or questions 🔷
🔷 **Reshare** to help others level up 🔷

Maciej Gomółka