Unit Testing in Swift

What to Test?

Core Functionality

Common UI Workflows (UI Testing)

Boundary Conditions

Bug Fixes

Criteria for Good Tests

Fast: Tests should run quickly

Independent: Different tests should not share state with each other

Repeatable: Results are the same every time a test is run

Self-validating: Either pass or fail with no interpretation required

Timely: Ideally, tests should be written before production code

Parts of a Unit Test

Typically 3 parts to a unit test:

1. **Given**
   a. Set up any values needed
2. **When**
   a. Execute the code being tested
3. **Then**
   a. Make an assertion about the results

Types of Assertions

Boolean: `XCTAssertTrue(), XCTAssertFalse()`

Nil: `XCTAssertNil(), XCTAssertNotNil()`

Equality: `XCTAssertEqual(), XCTAssertNotEqual()`

Compare: `XCTAssertGreaterThan(), XCTAssertLessThan()`

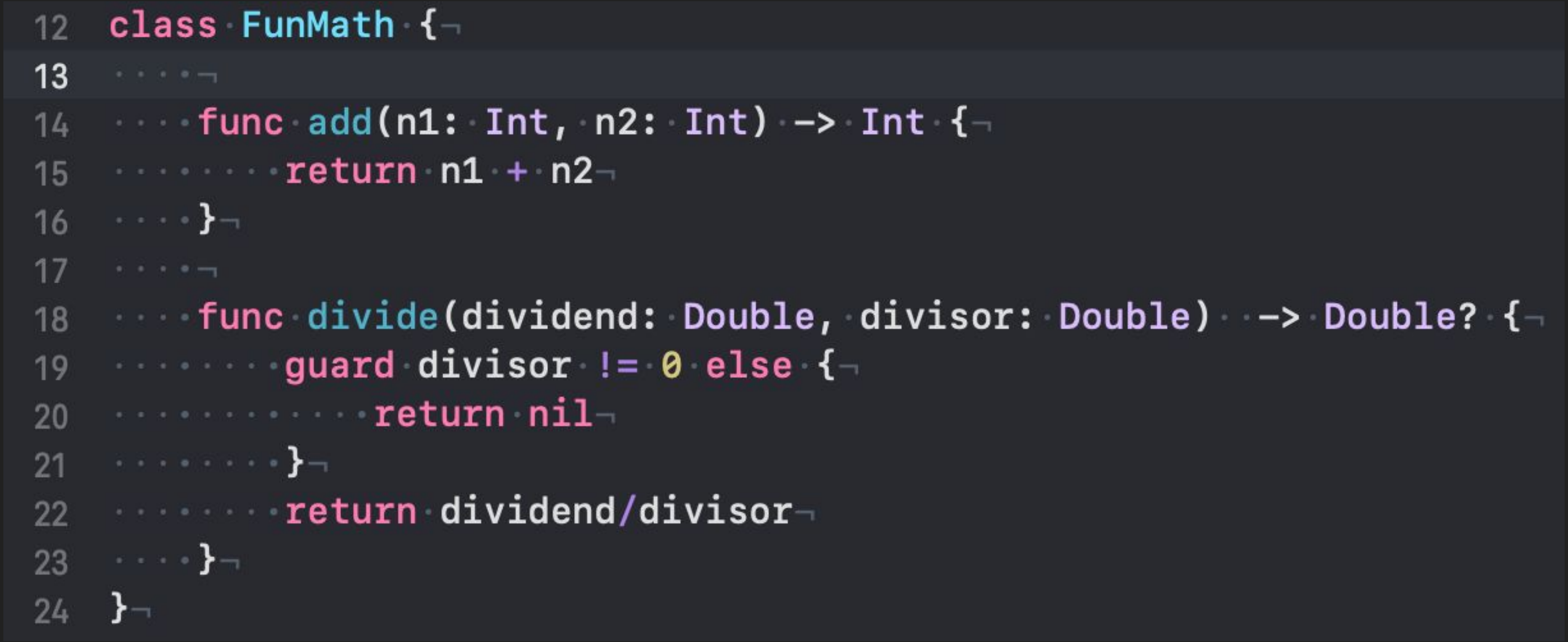
Error: `XCTAssertThrowsError(), XCTAssertNoThrow()`

Getting Started with iOS Testing

Four Main Steps

1. Import the testing target
2. Set up for test
3. Execute test(s) on the SUT
4. Clean up and tear down

Example SUT



```swift
class FunMath {
    
    func add(n1: Int, n2: Int) -> Int {
        return n1 + n2
    }
    
    func divide(dividend: Double, divisor: Double)  -> Double? {
        guard divisor != 0 else {
            return nil
        }
        return dividend/divisor
    }
}
```

What Should We Test?

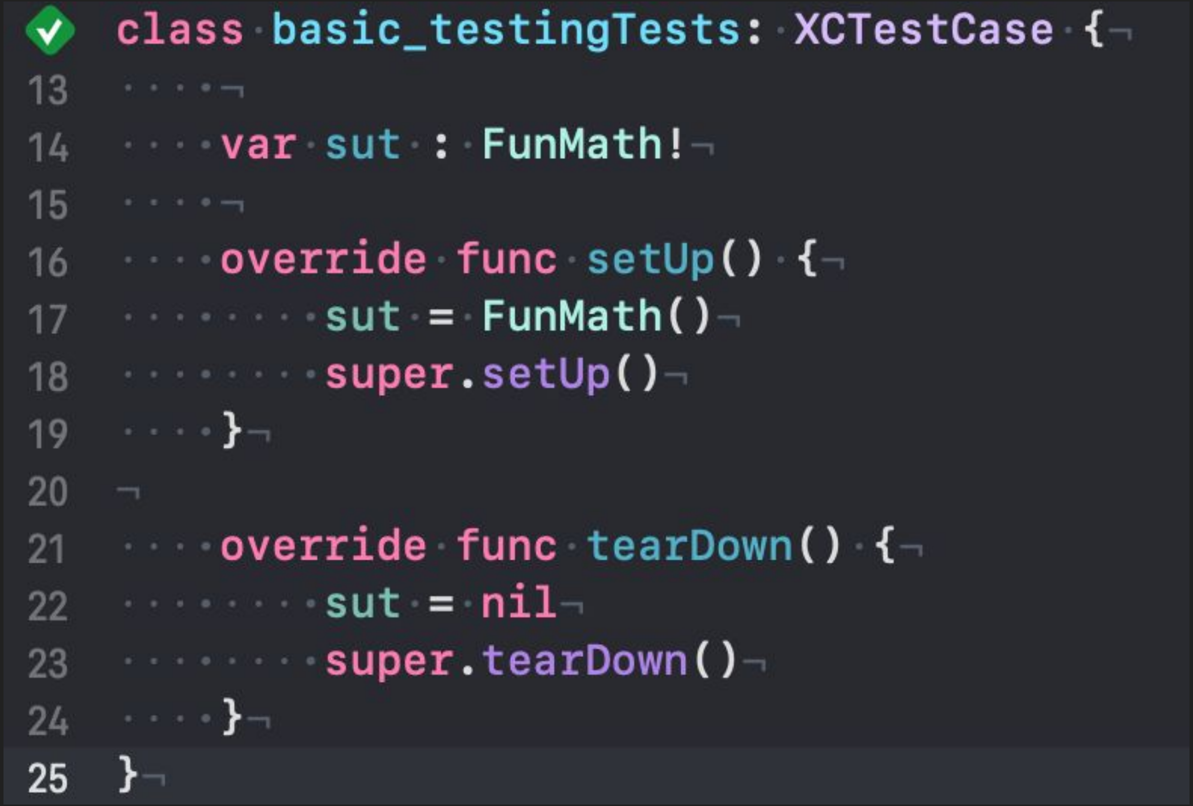
Core Functionality

- Make sure that the methods work with valid inputs

Boundary Case

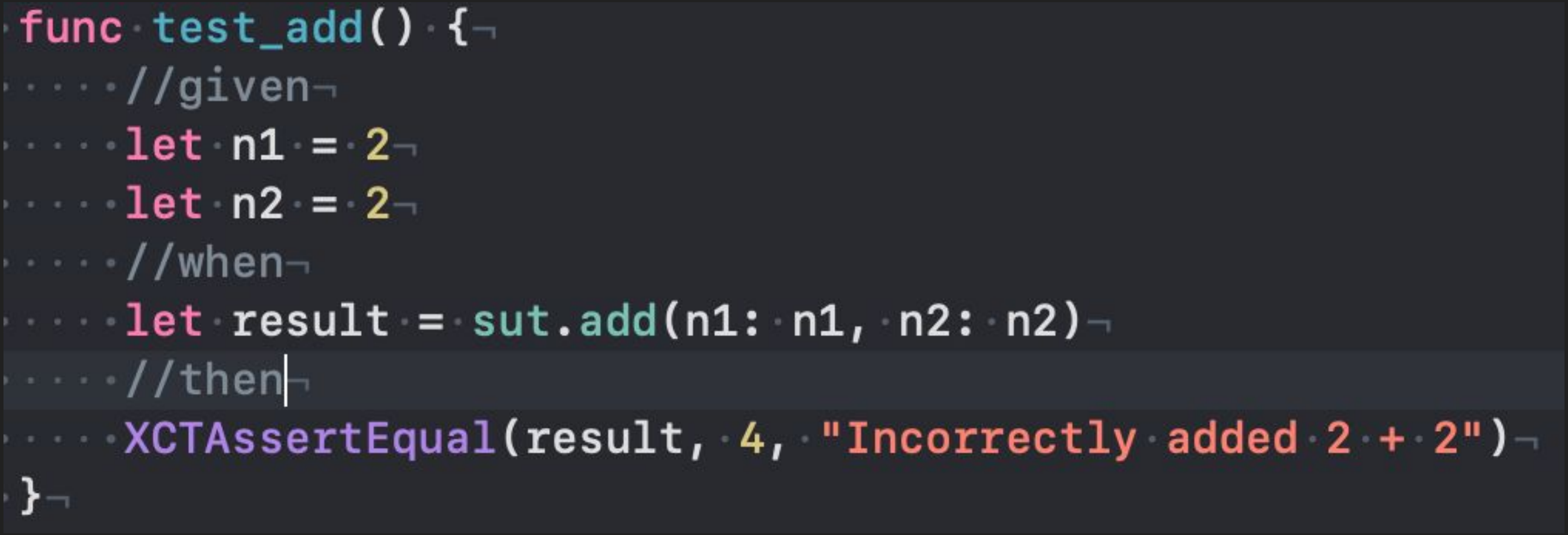
- Make sure that the divide() method returns nil if the divisor is 0

`setUp()` and `tearDown()`



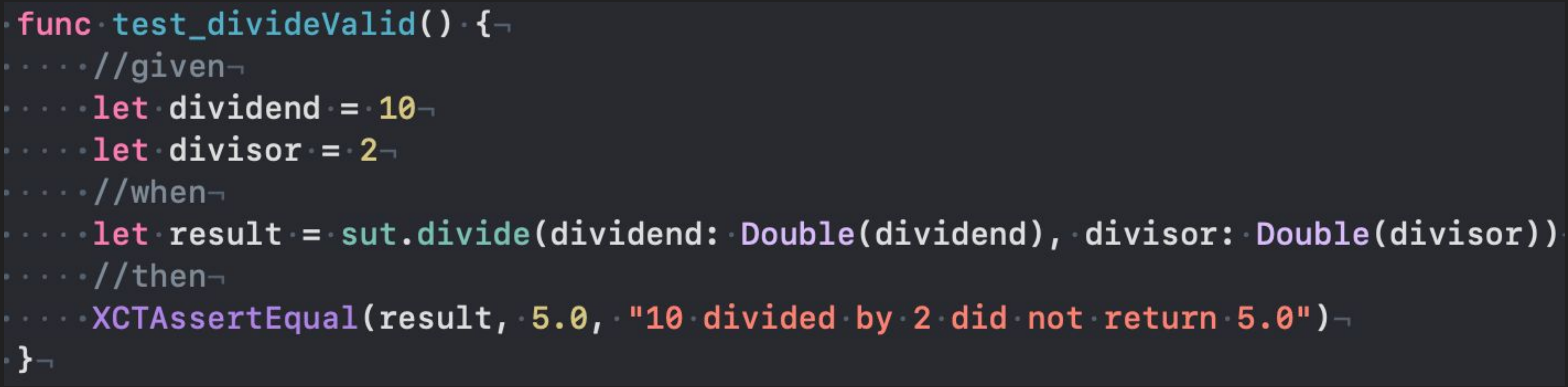
```swift
class basic_testingTests: XCTestCase {

    var sut : FunMath!

    override func setUp() {
        sut = FunMath()
        super.setUp()
    }

    override func tearDown() {
        sut = nil
        super.tearDown()
    }
}
```

Test the `add()` Method



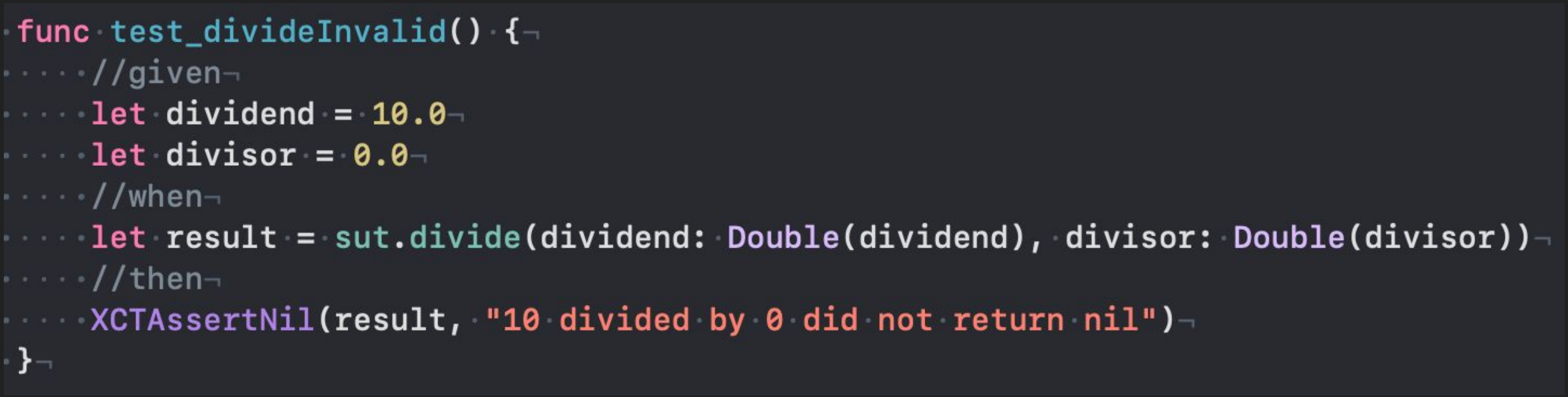
```
func test_add() {
    //given
    let n1 = 2
    let n2 = 2
    //when
    let result = sut.add(n1: n1, n2: n2)
    //then
    XCTAssertEqual(result, 4, "Incorrectly added 2 + 2")
}
```

Test Valid Use of `divide()`



```
func test_divideValid() {
    //given
    let dividend = 10
    let divisor = 2
    //when
    let result = sut.divide(dividend: Double(dividend), divisor: Double(divisor))
    //then
    XCTAssertEqual(result, 5.0, "10 divided by 2 did not return 5.0")
}
```

Test Invalid Use of `divide()`



```swift
func test_divideInvalid() {
    //given
    let dividend = 10.0
    let divisor = 0.0
    //when
    let result = sut.divide(dividend: Double(dividend), divisor: Double(divisor))
    //then
    XCTAssertNil(result, "10 divided by 0 did not return nil")
}
```