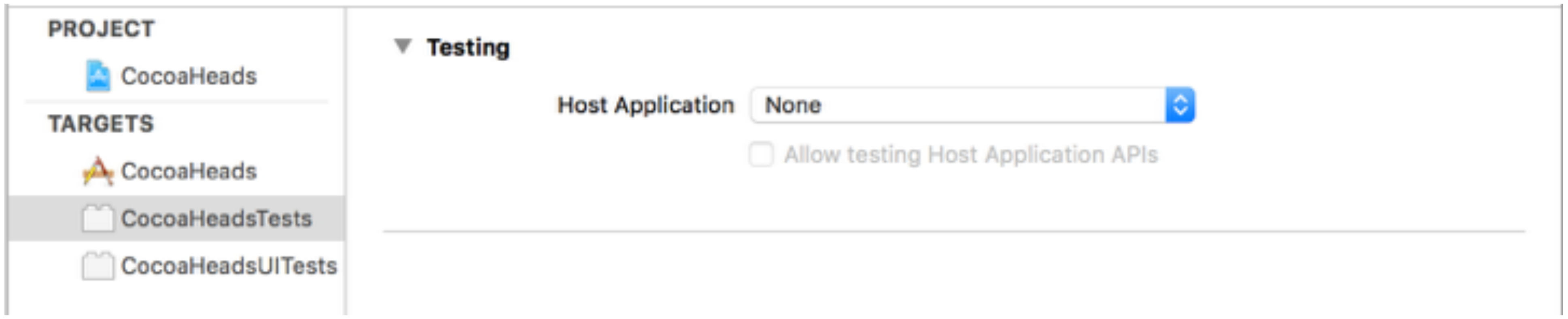
Stub Network Requests in UI Tests

Erik Sundin, Blocket
erik.sundin@blocket.se

Testing IOS Apps

- Unit Tests (Non-Hosted) VS Application Tests (Hosted)

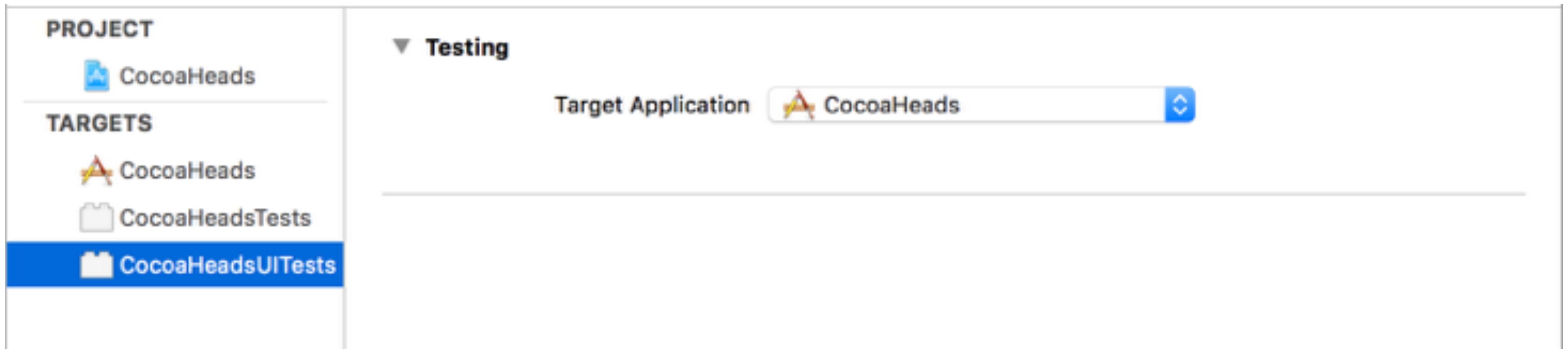


PROJECT
CocoaHeads
TARGETS
CocoaHeads
CocoaHeadsTests
CocoaHeadsUITests

▼ Testing

Host Application: None

Allow testing Host Application APIs



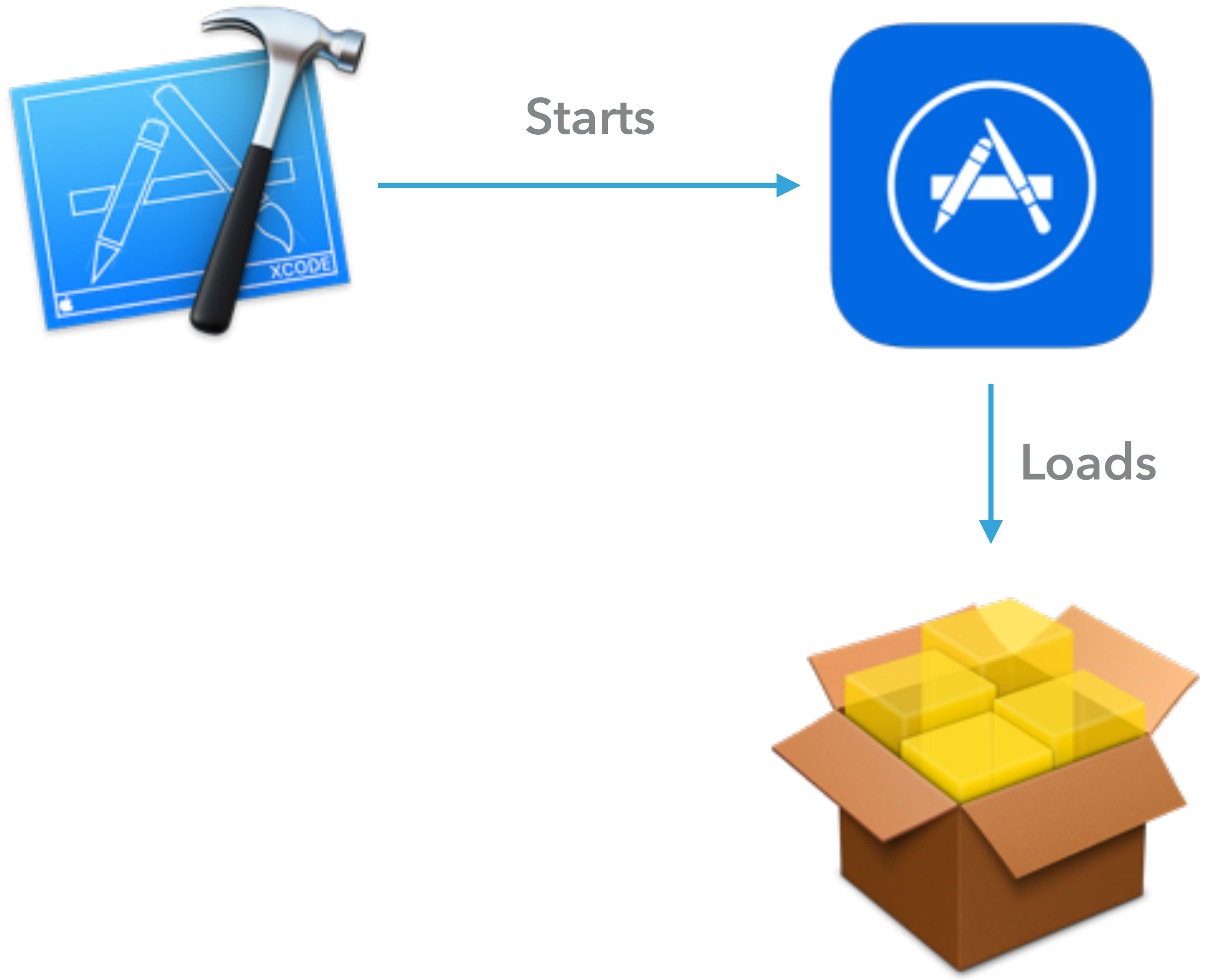
PROJECT
CocoaHeads
TARGETS
CocoaHeads
CocoaHeadsTests
CocoaHeadsUITests

▼ Testing

Target Application: CocoaHeads

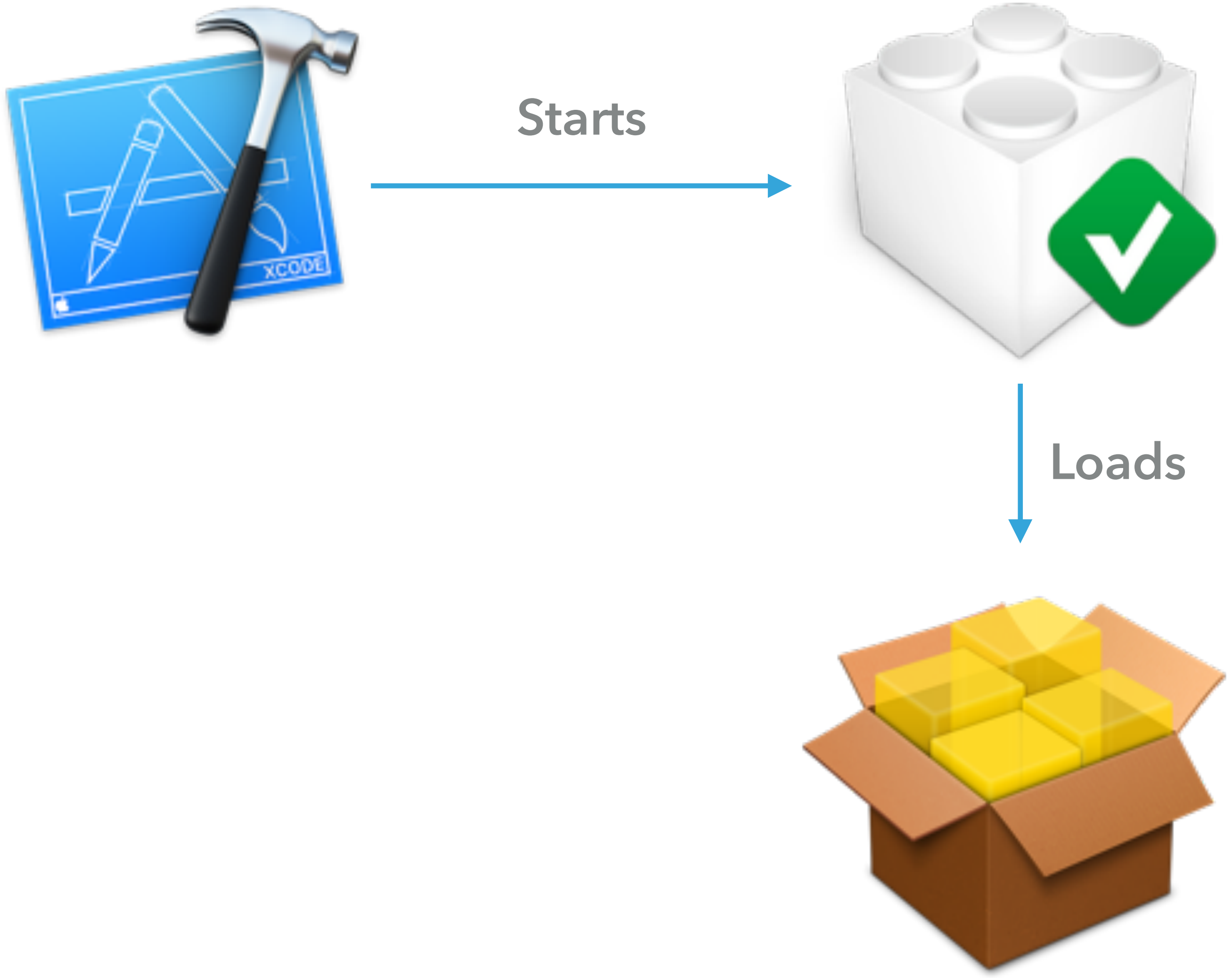
Testing IOS Apps

- Running an app



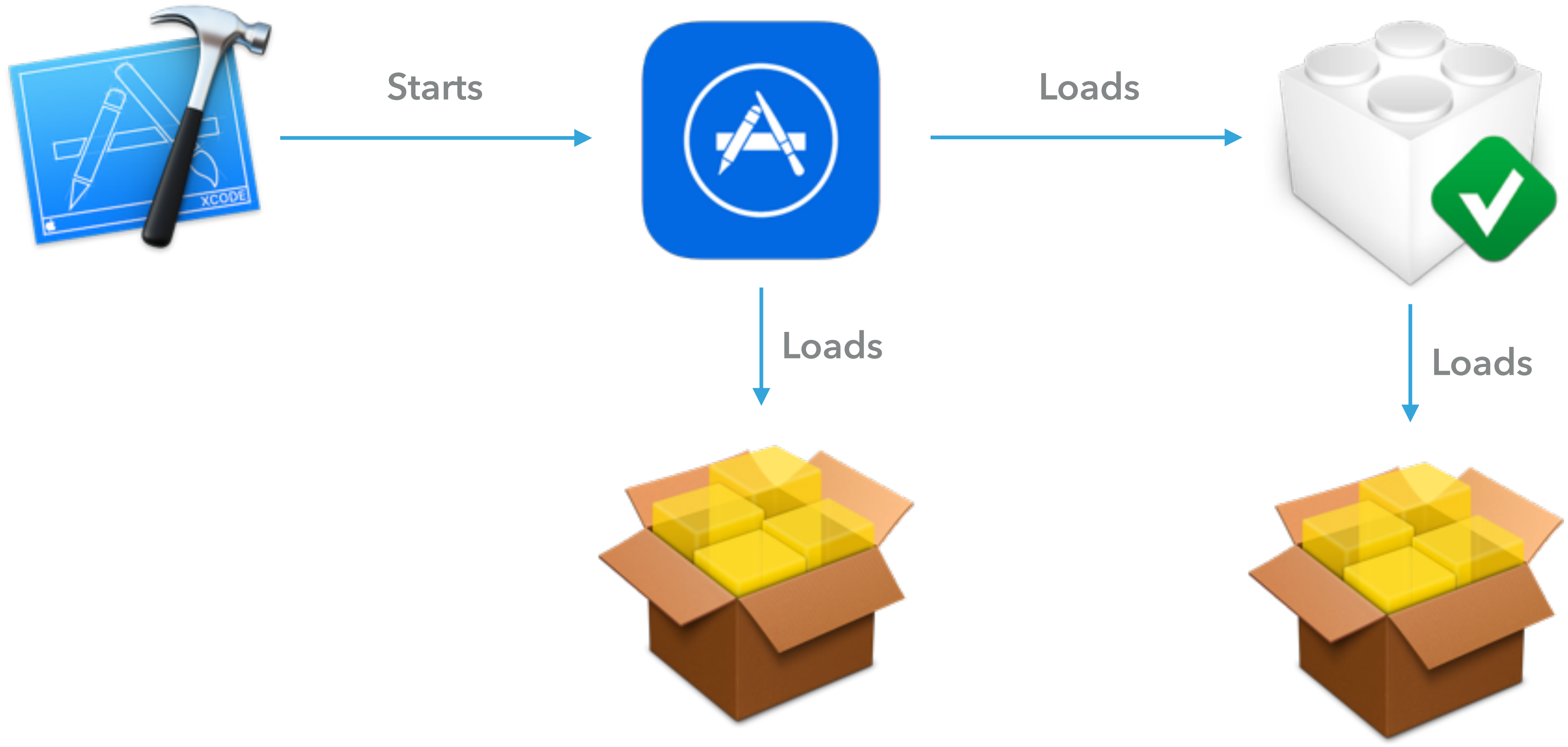
Testing IOS Apps

- Running a unit test (Non-Hosted)



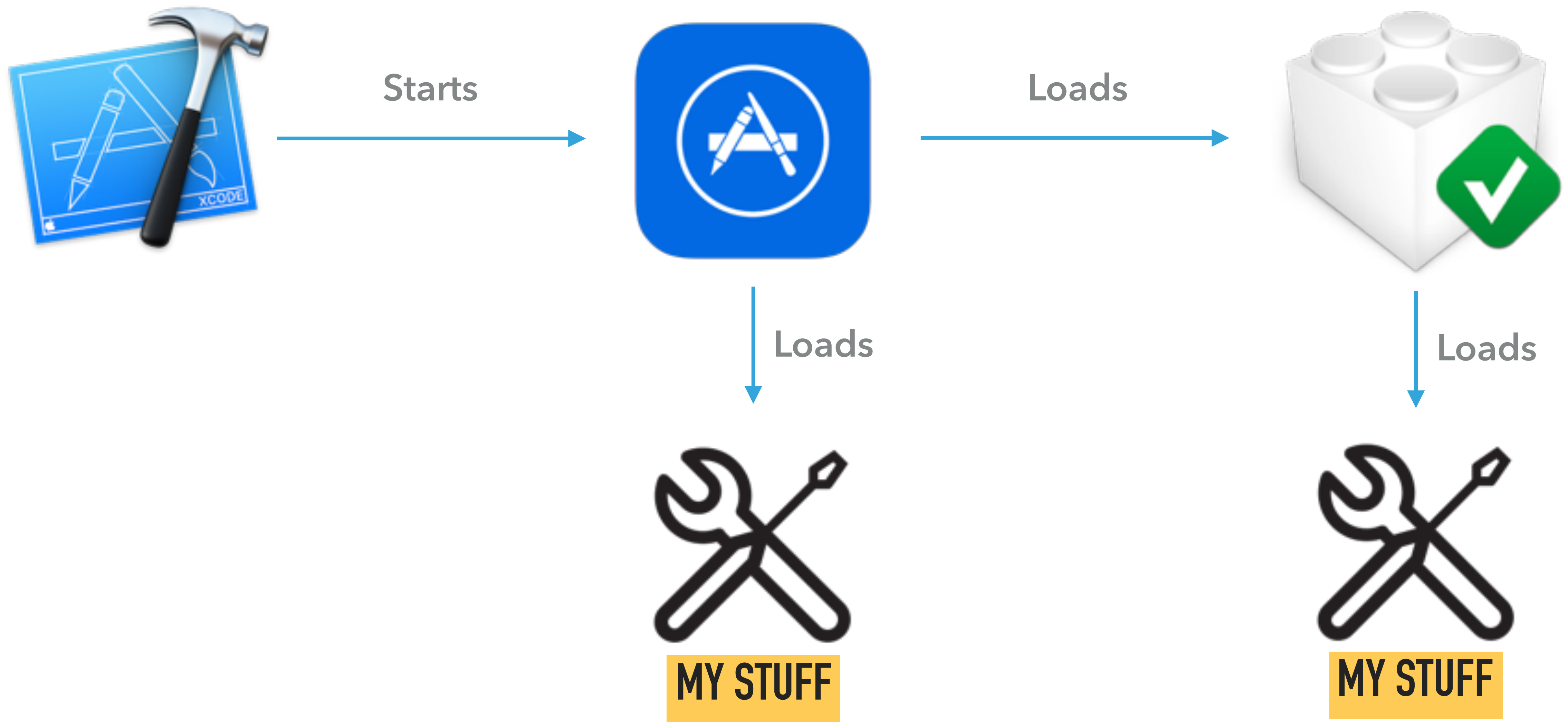
Testing IOS Apps

- Running an application test (Hosted)



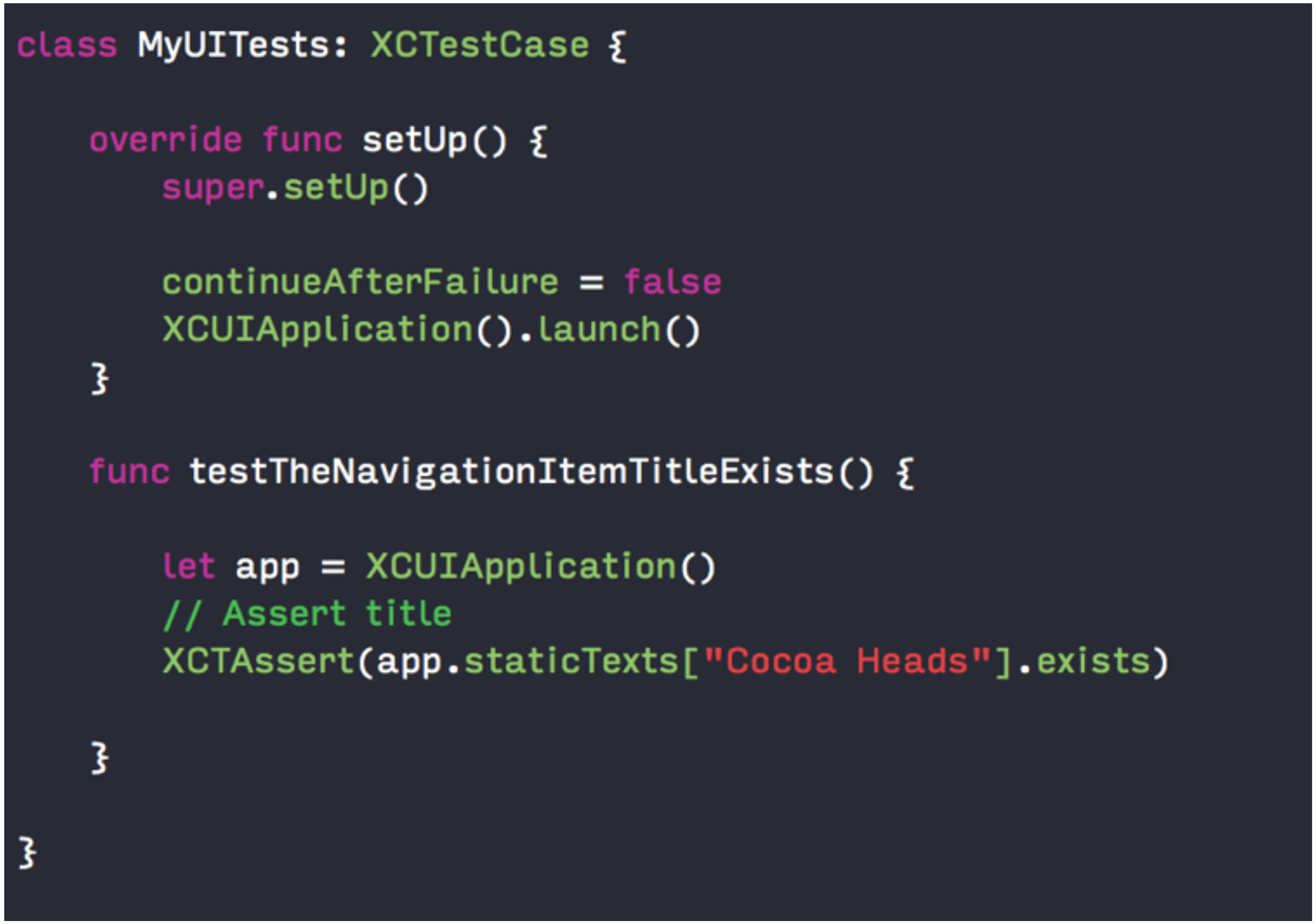
Testing IOS Apps

- Running an application test (Hosted)



FFFFFFFFFFFFFFFFFFFFFFFFFFFFUUUUUUUUUUUUUUUUUUUUUUUUUUUU

UI Tests



```
class MyUITests: XCTestCase {

    override func setUp() {
        super.setUp()

        continueAfterFailure = false
        XCUIApplication().launch()
    }

    func testTheNavigationItemTitleExists() {

        let app = XCUIApplication()
        // Assert title
        XCTAssert(app.staticTexts["Cocoa Heads"].exists)

    }

}
```

I want to run UI Tests with pre-defined data! That’s gonna be awesome!!!11

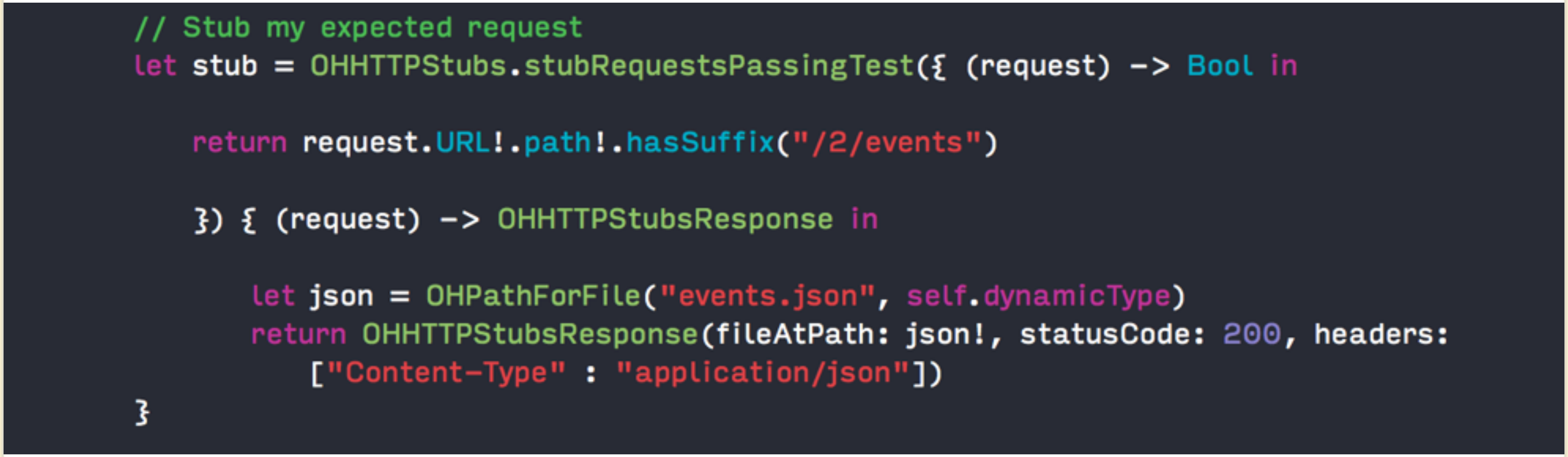
Developer

Lets use OHHTTPStubs, that works great for our other tests.

Developer

OHHTTPStubs

- https://github.com/AliSoftware/OHHTTPStubs

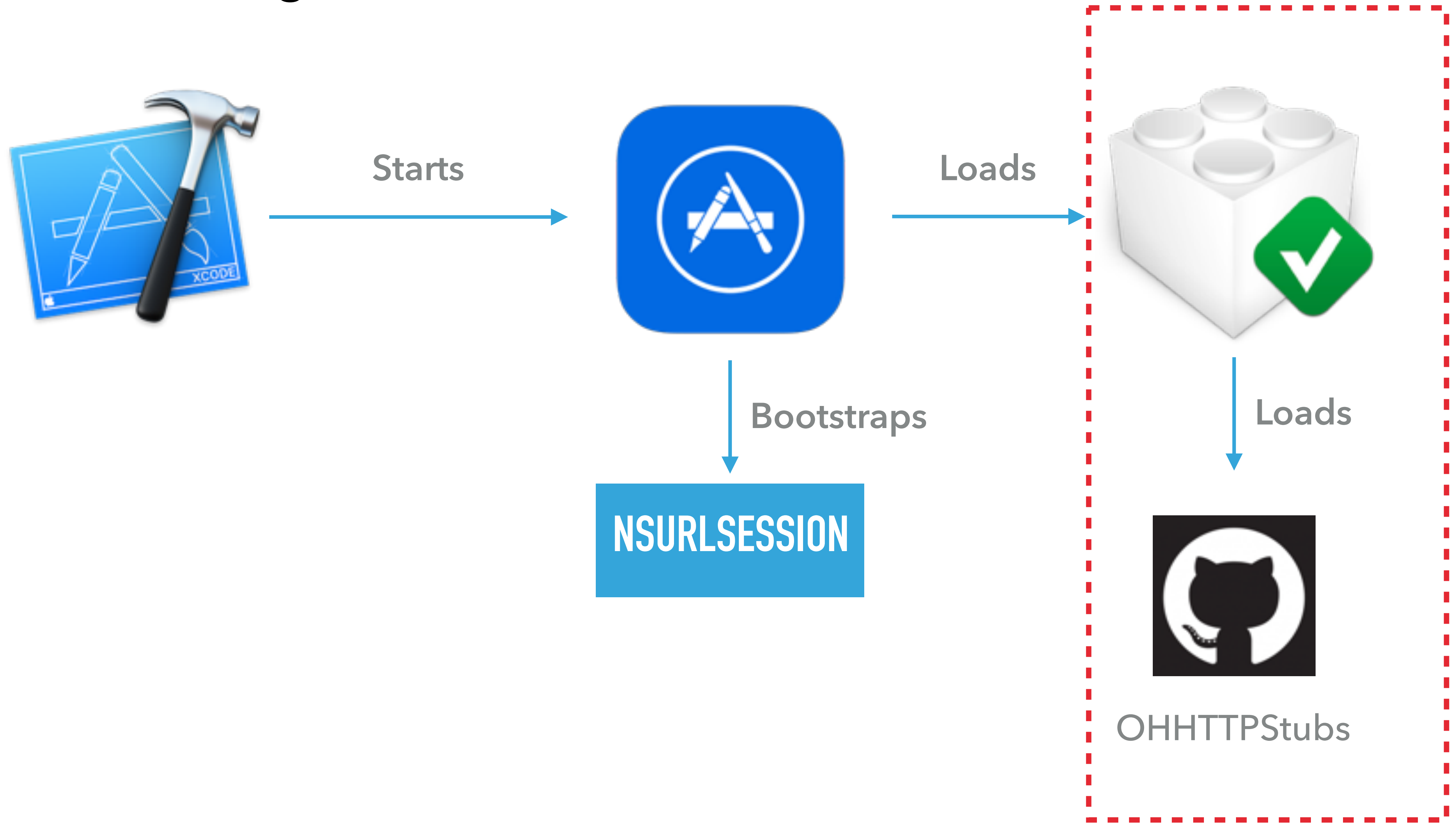


```swift
// Stub my expected request
let stub = OHHTTPStubs.stubRequestsPassingTest({ (request) -> Bool in

    return request.URL!.path!.hasSuffix("/2/events")

    }) { (request) -> OHHTTPStubsResponse in

        let json = OHPathForFile("events.json", self.dynamicType)
        return OHHTTPStubsResponse(fileAtPath: json!, statusCode: 200, headers:
            ["Content-Type" : "application/json"])
}
```

FFFFFFFFFFFFFFFFFFFFFFFFFFUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

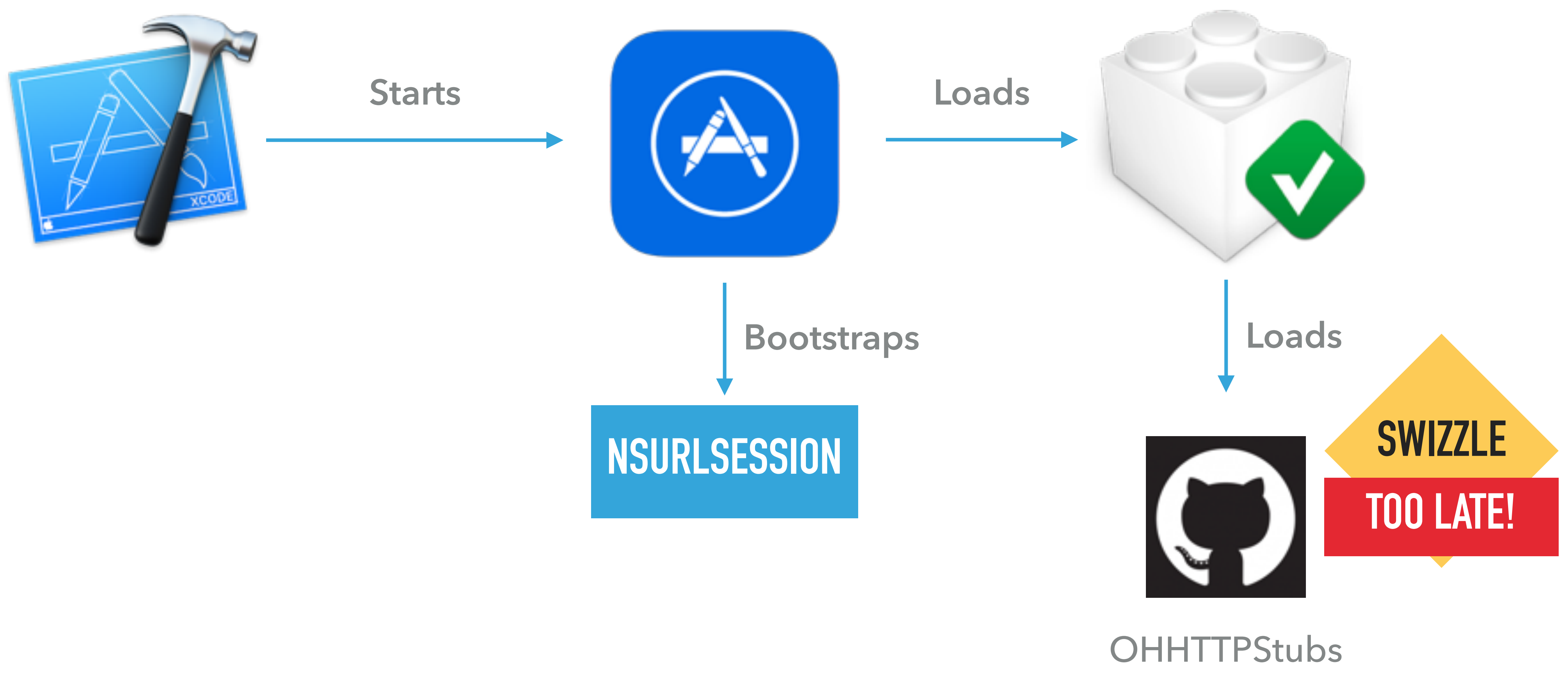
UI Tests are special

- Running



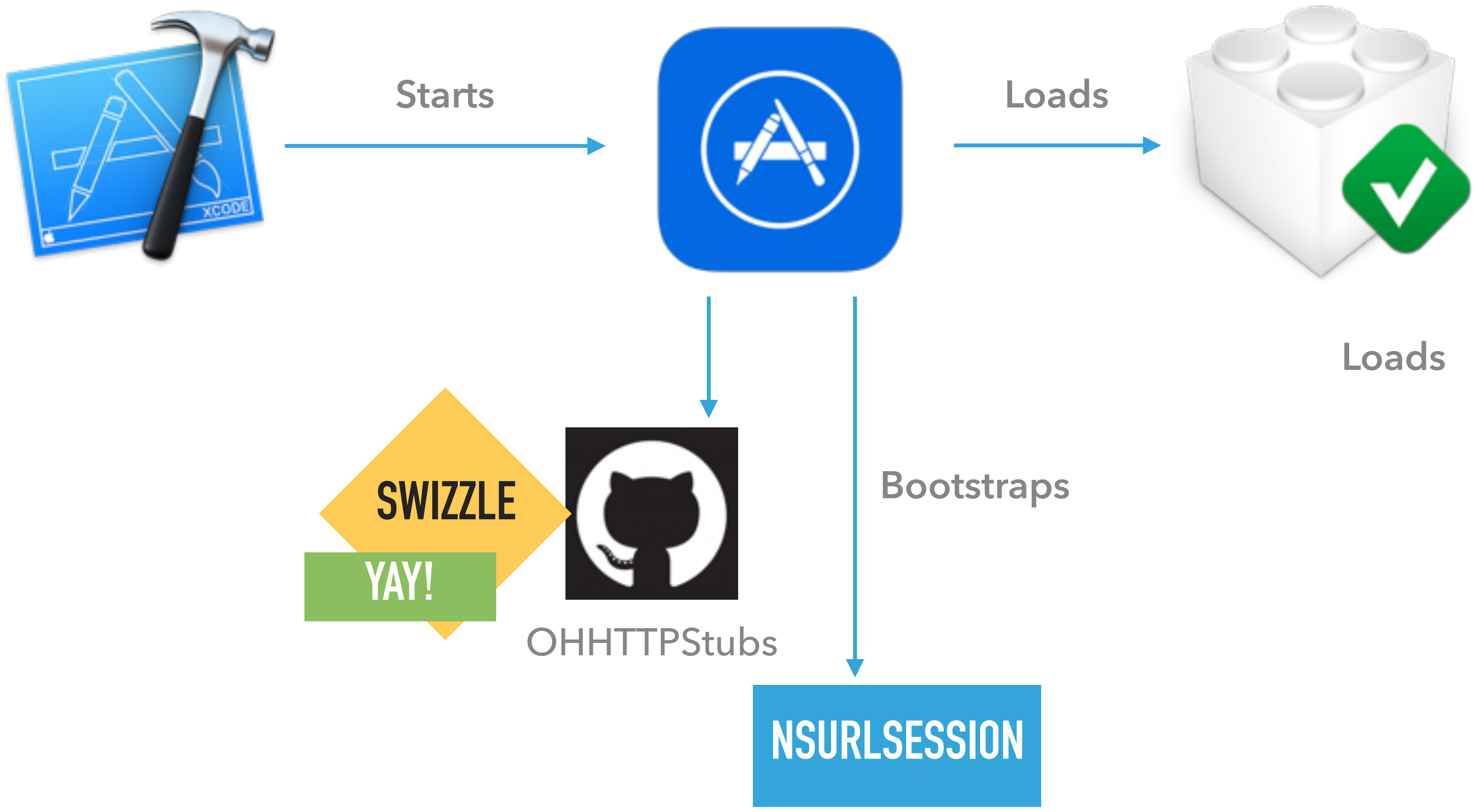
If UI Tests were normal

- Running

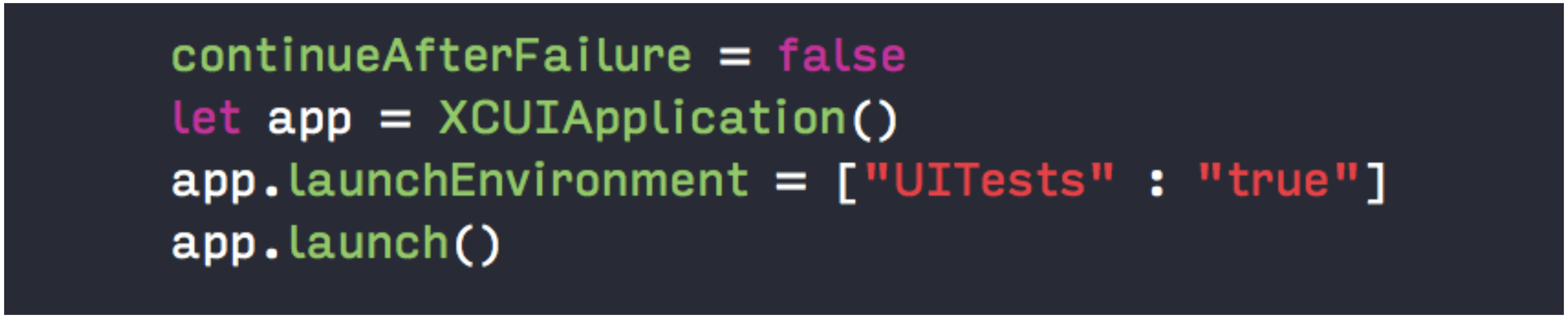


Swizzle Win

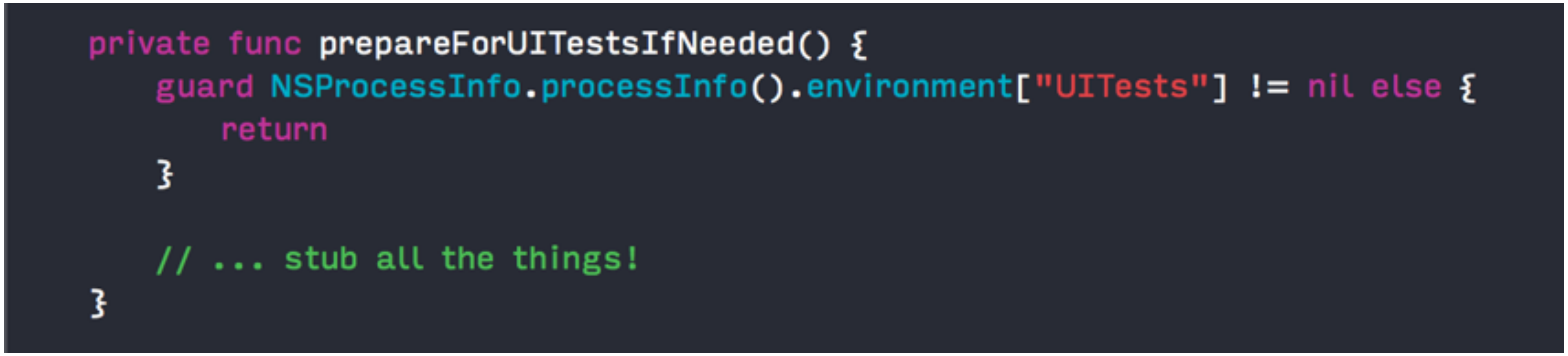
- Second try



Launch environment to the rescue



```
continueAfterFailure = false
let app = XCUIApplication()
app.launchEnvironment = ["UITests" : "true"]
app.launch()
```



```
private func prepareForUITestsIfNeeded() {
    guard NSProcessInfo.processInfo().environment["UITests"] != nil else {
        return
    }

    // ... stub all the things!
}
```

Why do we have a dependency on a stubbing framework in our app?

Annoying Colleague

FFFFFFFFFFFFFFFFFFFFFFFFFFFFUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

Drawbacks

- Dependencies and resources that shouldn’t be there
- Logic for setting up tests in production app code
- Testing logic in multiple places

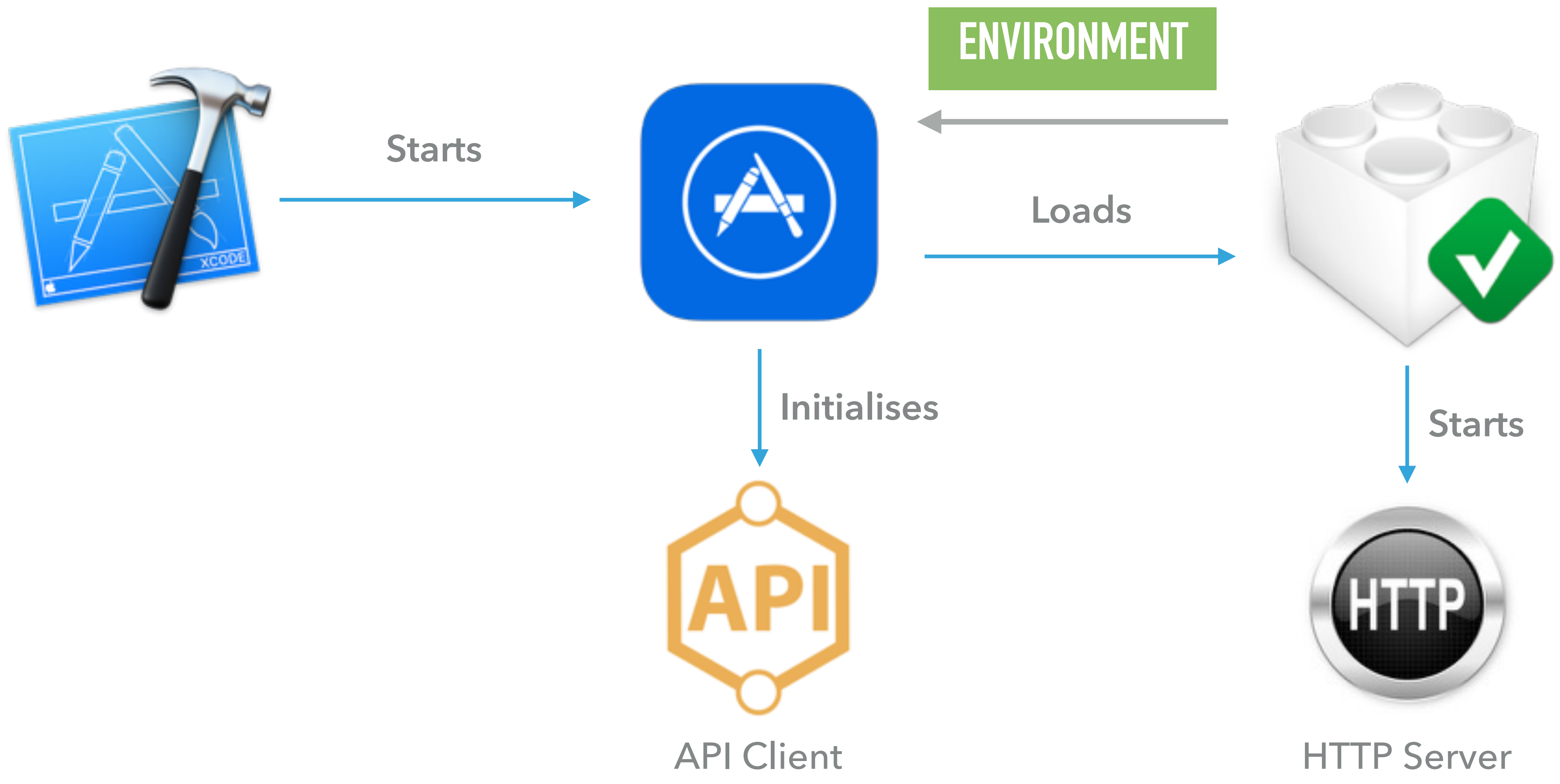
Let’s try a different approach

A Test Contained HTTP server

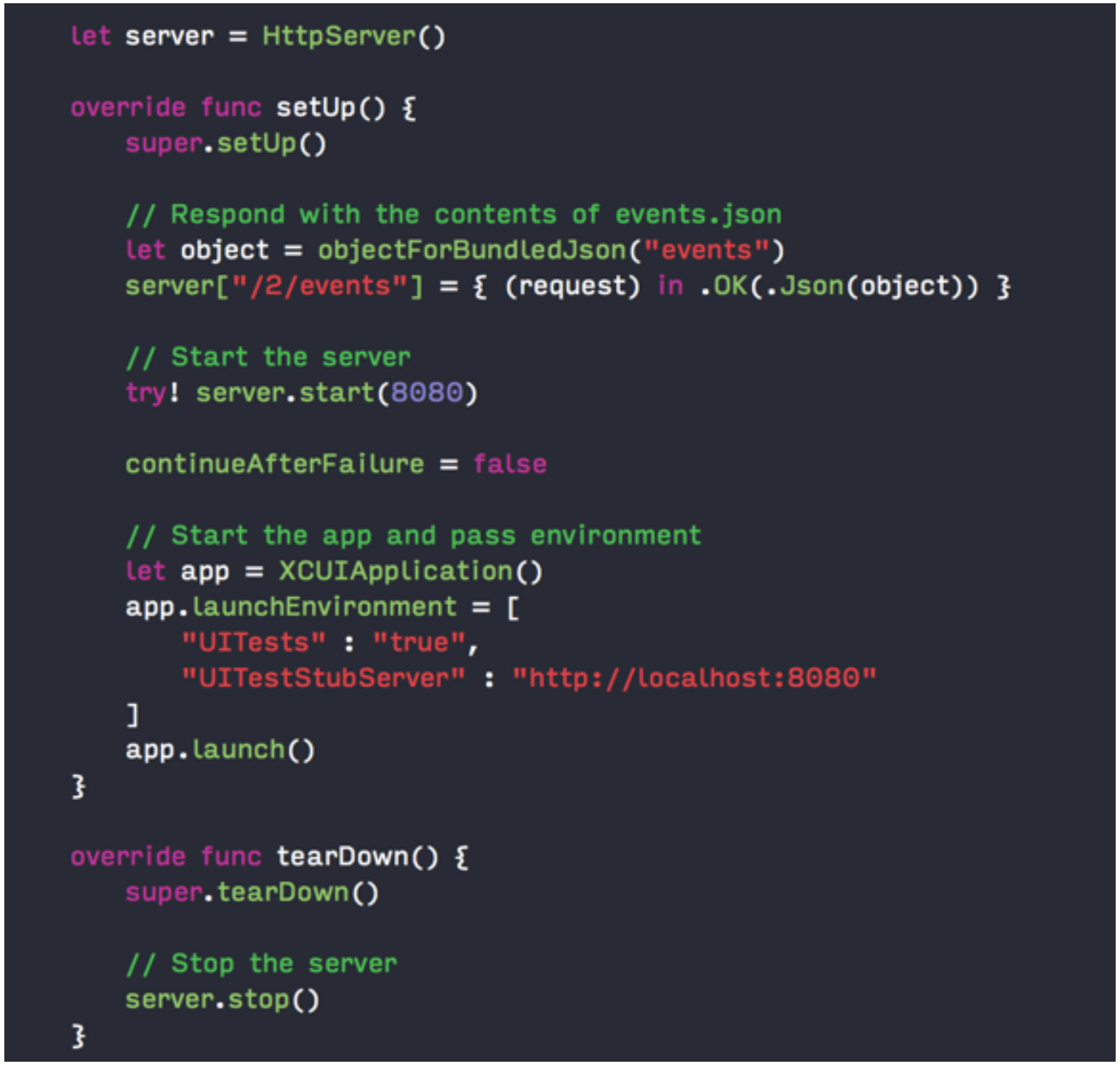
- Minimal impact on app code
- No testing dependencies / stubbing logic in the app
- Flow of the test is contained within the UITest

A different approach

- A test contained HTTP Server



https://github.com/httpswift/swifter



```swift
let server = HttpServer()

override func setUp() {
    super.setUp()

    // Respond with the contents of events.json
    let object = objectForBundledJson("events")
    server["/2/events"] = { (request) in .OK(.Json(object)) }

    // Start the server
    try! server.start(8080)

    continueAfterFailure = false

    // Start the app and pass environment
    let app = XCUIApplication()
    app.launchEnvironment = [
        "UITests" : "true",
        "UITestStubServer" : "http://localhost:8080"
    ]
    app.launch()
}

override func tearDown() {
    super.tearDown()

    // Stop the server
    server.stop()
}
```

Let’s have a look at some code

Where to go from here

- Easier to build out UI Tests
- No need to maintain a dedicated test environment or such
- Automated screenshots with snapshot (Fastlane)

Stub Network Requests in UI Tests

Erik Sundin, Blocket
erik.sundin@blocket.se