Unit Testing on the Mac

Daniel Jalkut
Red Sweater Software

red sweater

Who Cares?

- People are passionate about unit testing
- Decide for yourself by understanding:
  - What unit testing is...
  - What are the costs and benefits...
  - How much work for your Mac project...

What are Tests?

- A critical examination, observation, or evaluation
- Code that confirms an expected behavior of some other code:

```
if ((GetPI() < 3.14) || (GetPI() > 3.14))
{
   printf("Oh crap, we're screwed.\n");
}
```

What are Unit Tests?

- "Just Code"
- Run systematically
- Ideal properties
  - Fast execution
  - Small scope
  - Limited external dependencies

Example: Does my addition function produce the expected answer "7" for the question "3 + 4" ?

Integration Tests

- Larger scope
- May use variable inputs (fuzzing!)
- May use external dependencies
- May take longer to run

Example: "Does my addition function produce similar results to Google's calculator, for a collection of fixed and randomized inputs?"

red sweater

Why Test?

- Lock-in correct behavior
- Document your code base
- Inspire a cleaner architecture
- Confidence!

MANTRA: "Trust, but verify!"

red sweater

Why Skip it?

- It takes time
- Some code bases are particularly hard
- May provide false confidence
- Because Wil Shipley says so

Testing with Xcode

- OCUnit is bundled with Xcode
- Create a unit testing bundle target
- Write concrete subclasses of SenTestCase
- Link to your product directly or to a subset of your app's source files

SenTestCase Subclasses

- The magic of test case classes
- For every method starting with “test”...
  - A setup method is run.
  - The test method itself is run.
  - A tearDown method is run.

Be More Assertive!

- Any failed assertion triggers a build error
- A variety of assertion macros are available:

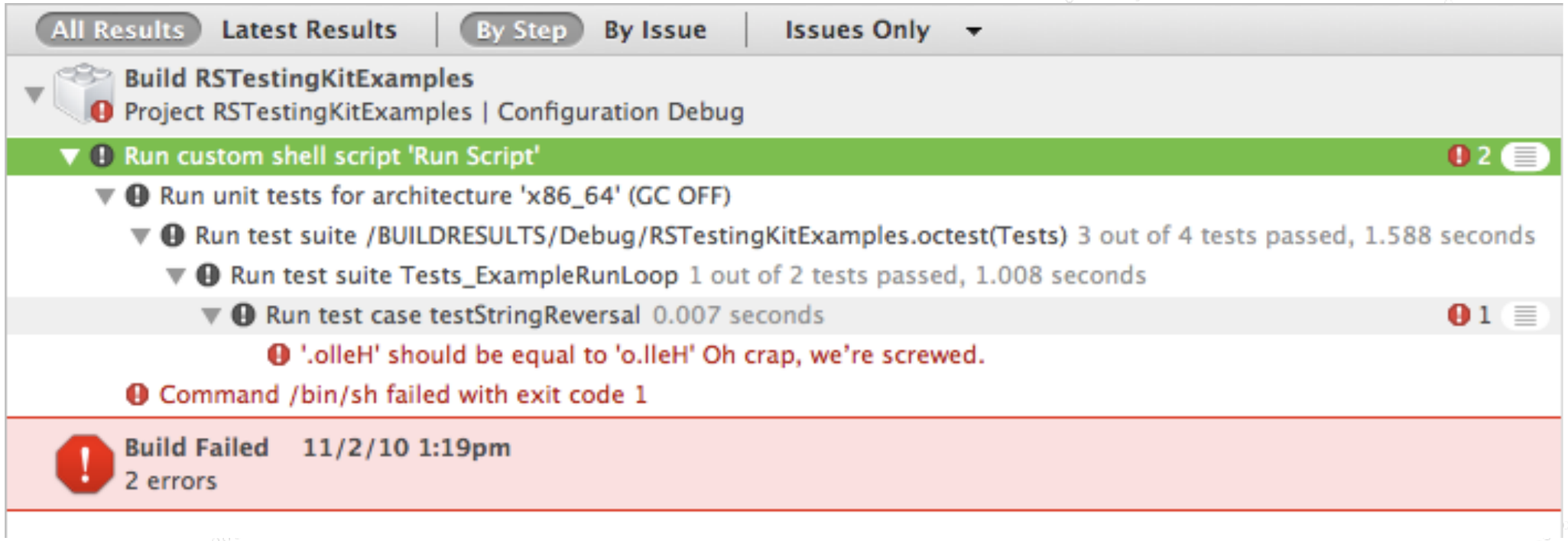
```
STAssertTrue(NO, @"Sabotaged this test!");

STFail(@"This one, too!");

STAssertEqualsWithAccuracy(0.95, 1.05, 0.1, @"Close enough!");
```

A Simple Test

```
@implementation MyTestCase

- (void) testStringReversal
{
    NSString* helloString = @"Hello.";
    NSString* expectedString = @".olleH";
    NSString* actualString = nil;

    // Reverse the string
    actualString = [helloString stringByReversing];

    // Confirm expectation
    STAssertEqualObjects(expectedString, actualString,
            @"Oh crap, we're screwed.");
}

@end
```

red sweater

When Things Go Wrong...



red sweater

What should you test?

From the start...

- Test-driven development
- Writing complex or hard to reason code
- Lock-in contractual expectations

After-the-fact...

- Bug cases as you fix them
- Exploring and confirming code behavior
- Code coverage

Un-Perfecting Unit Tests

- Some testing better than no testing
- Common issues I have run into:
  - Faking dependencies
  - Packaging input materials
  - Testing asynchronous code
  - Simulating network behavior

Fakes, Mocks and Hacks

- Fakes mimic the interface of a real object
- Mocks confirm interactions with a fake object
- OCMock is one solution, but not bundled
- Hacks get the job done
  - Methods designed for tests only
  - Pre-processor macros to alter behavior
  - Etc., Etc.

Packaging Inputs

- Some tests require cumbersome data
- Difficult to encode directly into source files
- Solution: Copy them into the test bundle

RSTestCase

- Subclass of SenTestCase
- Standardizes loading of bundled data
- Loads strings or data by simple filename
- Name of input folder can be overriden

```
testData = [self dataFromTestInputFilename:@"Test1"];
[self performTestWithData:testData];
```

Get RSTestingKit! http://bit.ly/RSTestingKit

red sweater

Asynchronous Code

- Some code paths involve a run loop callback
- Test cases expect to be run serially
- Solution: Codify a start/wait mechanism

Get RSTestingKit! http://bit.ly/RSTestingKit

red sweater

RSRunLoopWaitingTestCase

- Subclass of RSTestCase
- Spins RunLoop until completion criteria are met

```
[self setTimeoutFailureString: @"Waiting for Godot"];

[self startWaitingForGodot];

[self waitForRunLoopTestCompletion];

STAssertTrue(mGodotIsHere, @"Godot should be here.");
```

Get RSTestingKit! http://bit.ly/RSTestingKit

redsweater

HTTP Client Code

- Code relies on web-based services
- Behavior determined by web responses
- Solution: Simulate an HTTP network service

Get RSTestingKit! http://bit.ly/RSTestingKit

red sweater

RSHTTPClientTestCase

- Subclass of RSRunLoopWaitingTestCase
- Sends a request and waits for synthetic reply
- Allows you to colocate the code for both the request and response code.

Get RSTestingKit! http://bit.ly/RSTestingKit

red sweater

Request Kick-off

```
- (void) testSomeHTTPThing
{
    NSURL* simpleHTTPRequestURL =
       [self serverURLForHTTPTestNamed:@"SimpleHTTPRequest"];

    // Kick-off some network request with simpleHTTPRequestURL
    ...

    // Wait for completion
    [self waitForRunLoopTestCompletion];

    // Define STAssertions that validate result
    ...
}
```

Get RSTestingKit! http://bit.ly/RSTestingKit

red sweater

Response Synthesis

```
- (GTMHTTPResponseMessage *)
    responseForTestRequest_SimpleHTTPRequest:
    (GTMHTTPRequestMessage *)request
{
    // Do anything to fabricate an HTTP response for the test
    return [GTMHTTPResponseMessage
        responseWithHTMLString:@"<hello>"];
}
```

Get RSTestingKit! http://bit.ly/RSTestingKit

red sweater

In Summary

- Consider testing!
- For increased confidence in your code
- Don't settle for limitations of Apple's tools
- Break the "rules" as much as necessary to get the job done.

red sweater

One More Thing.

Daniel Jalkut
jalkut@red-sweater.com
www.red-sweater.com

Find Me on Twitter: @danielpunkass

Get RSTestingKit! http://bit.ly/RSTestingKit