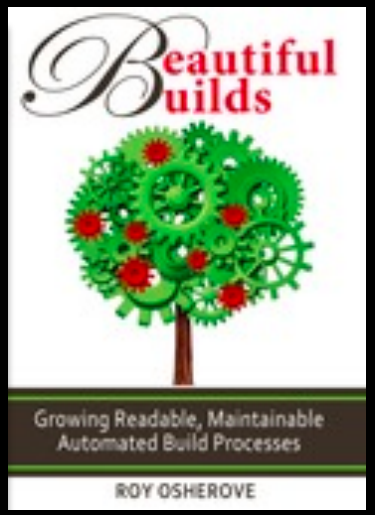
Unit Testing Best Practices and Horrible Mistakes

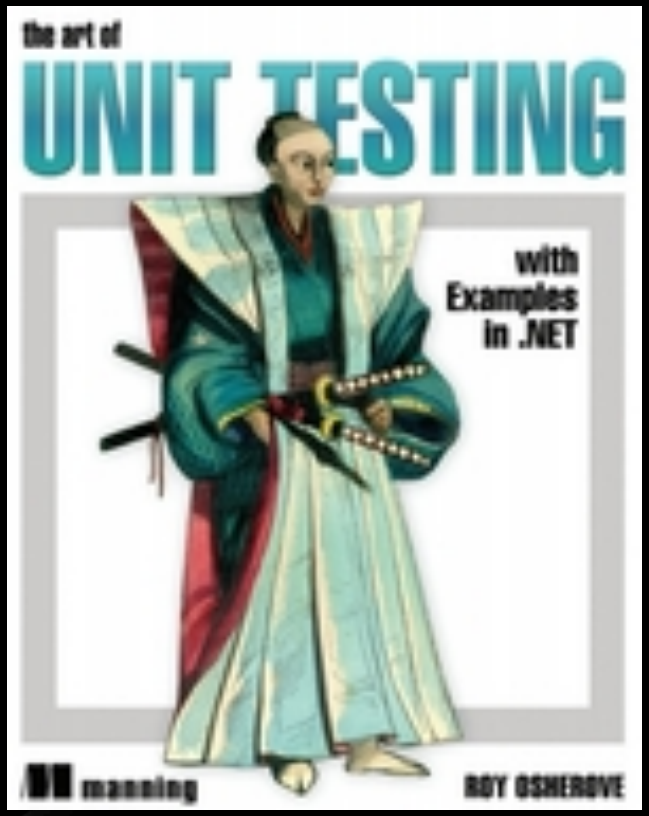
In JS

@RoyOsherove
ArtOfUnitTesting.com

Unit Testing Good Practices and Possible Mistakes

In JS





ArtOfUnitTesting.com
@RoyOsherove

Fail-First Experience

Just by Doing It, Unit Testing Makes Your Life Easier

Unit Testing makes your developer lives easier

- Easier to find bugs
- Easier to maintain
- Easier to understand
- Easier to **Develop**

No Test Review

Test review vs. code review

- Understand intent of developer
- 10 times quicker
- Drill in when needed

- Important for learning teams
- Helps drive change

Real World Agenda

- Test Reviews

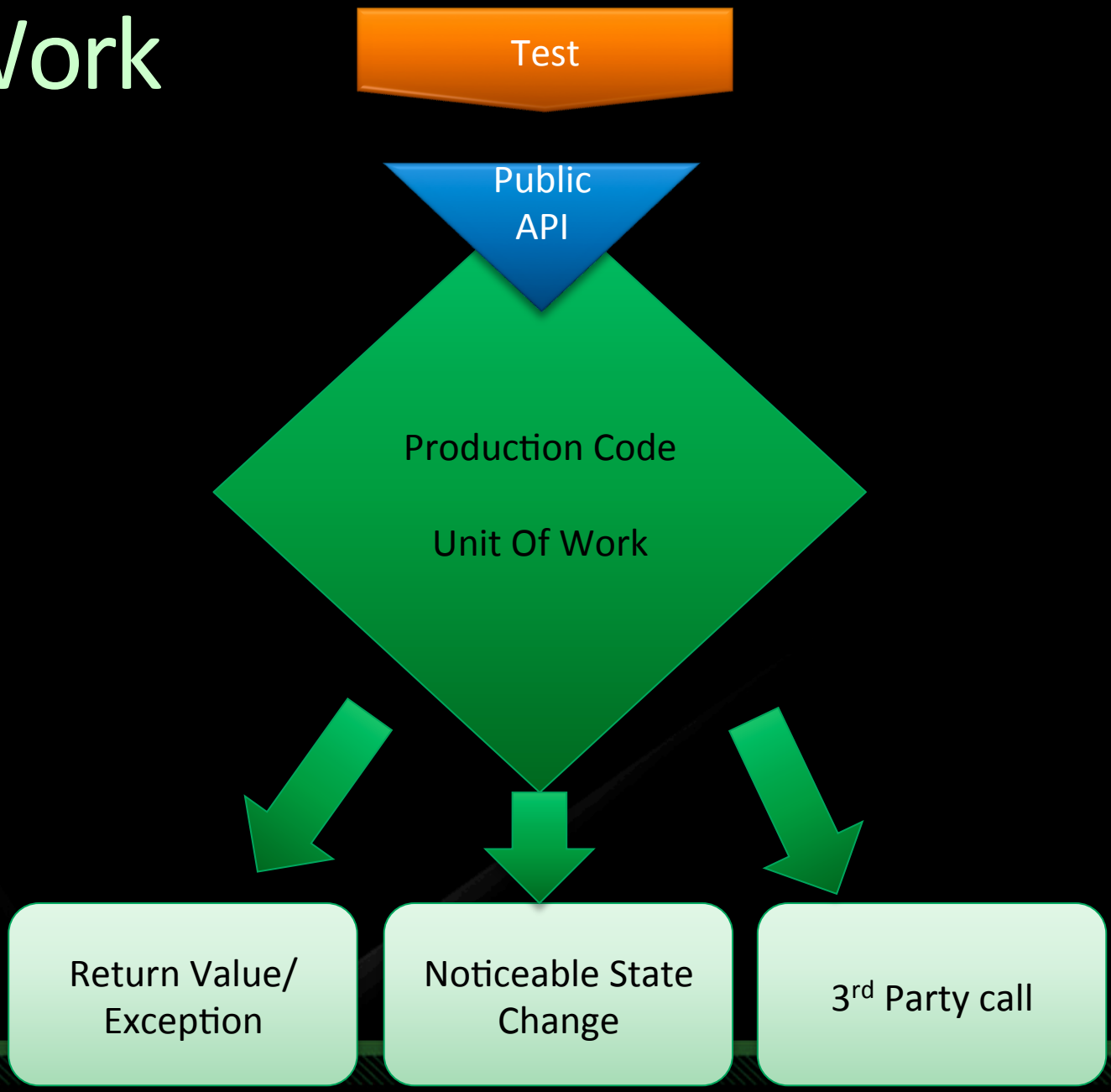
- Making your tests **TRUSTworthy**
- Creating **MAINTAINable** tests
- **READable** tests

RTFM

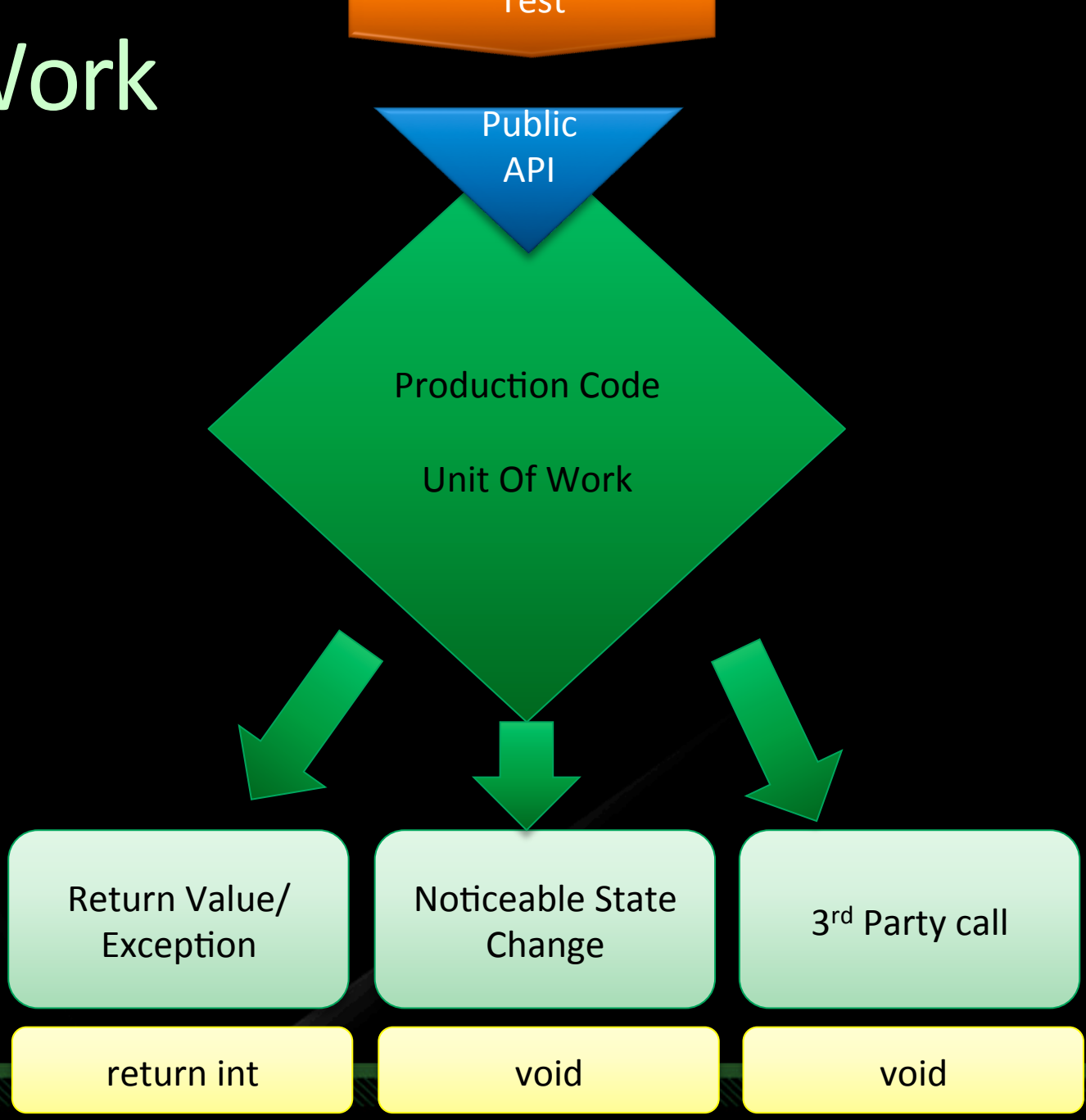
Faking
~~*Mocking*~~ all the things

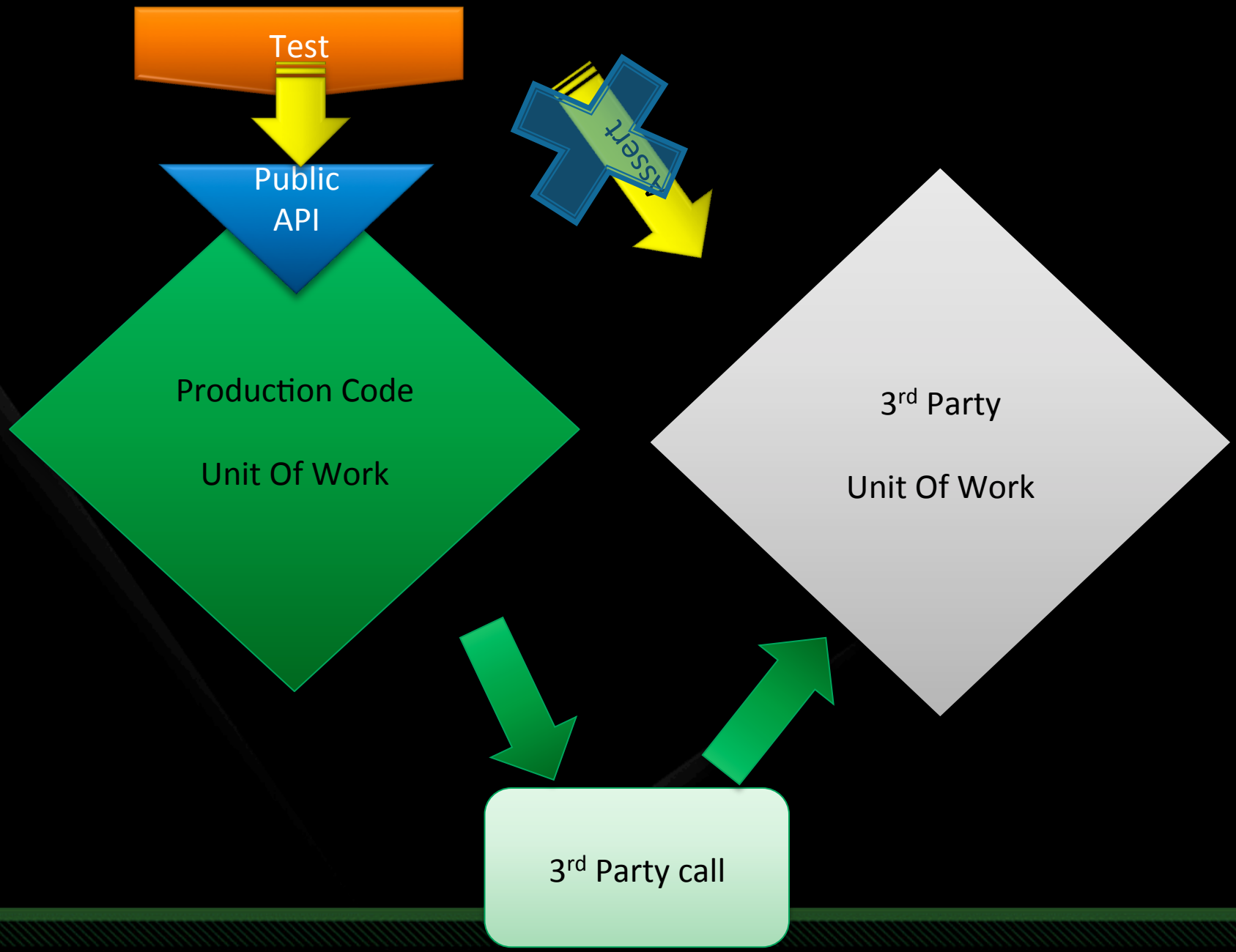
#Tests with mock objects
.should() < 5%

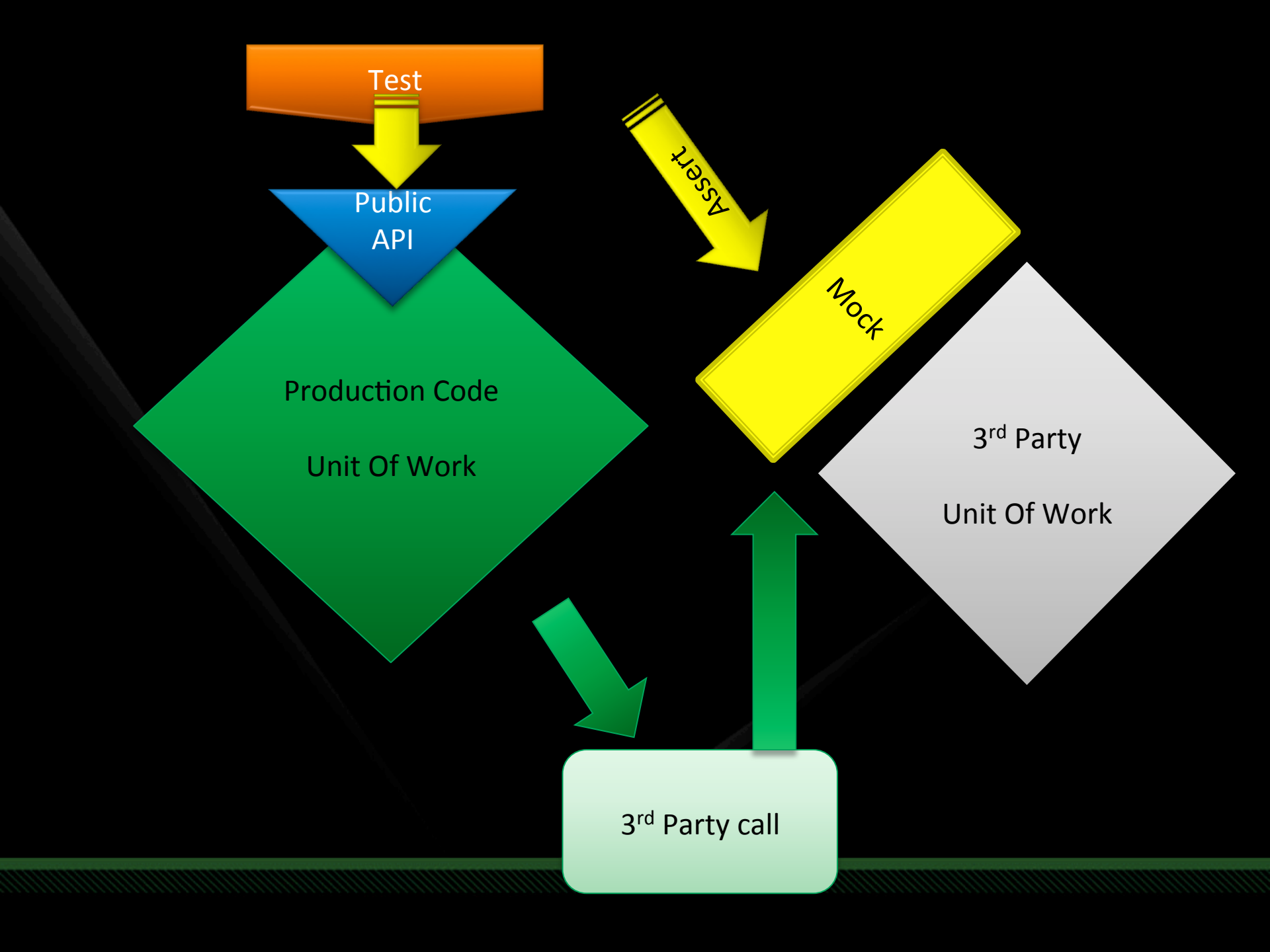
Unit Of Work


Test
Public API
Production Code
Unit Of Work
Return Value/ Exception
Noticeable State Change
3rd Party call

Unit Of Work


Test
Public API
Production Code
Unit Of Work
Return Value/ Exception
Noticeable State Change
3rd Party call
return int
void
void


Test
Public API
Assert
Production Code
Unit Of Work
3rd Party
Unit Of Work
3rd Party call


Test
Public API
Production Code
Unit Of Work
Assert
Mock
3rd Party
Unit Of Work
3rd Party call

Unit Of Work

5%

Return Value/ Exception

Noticeable State Change

3rd Party call

Everything is a mock

A simpler terminology

“Xunit Test Patterns”

- **Old**
- *Test Spy*
- *Test Double*
- *Fake*
- *Mock*
- *Stub*

A simpler terminology

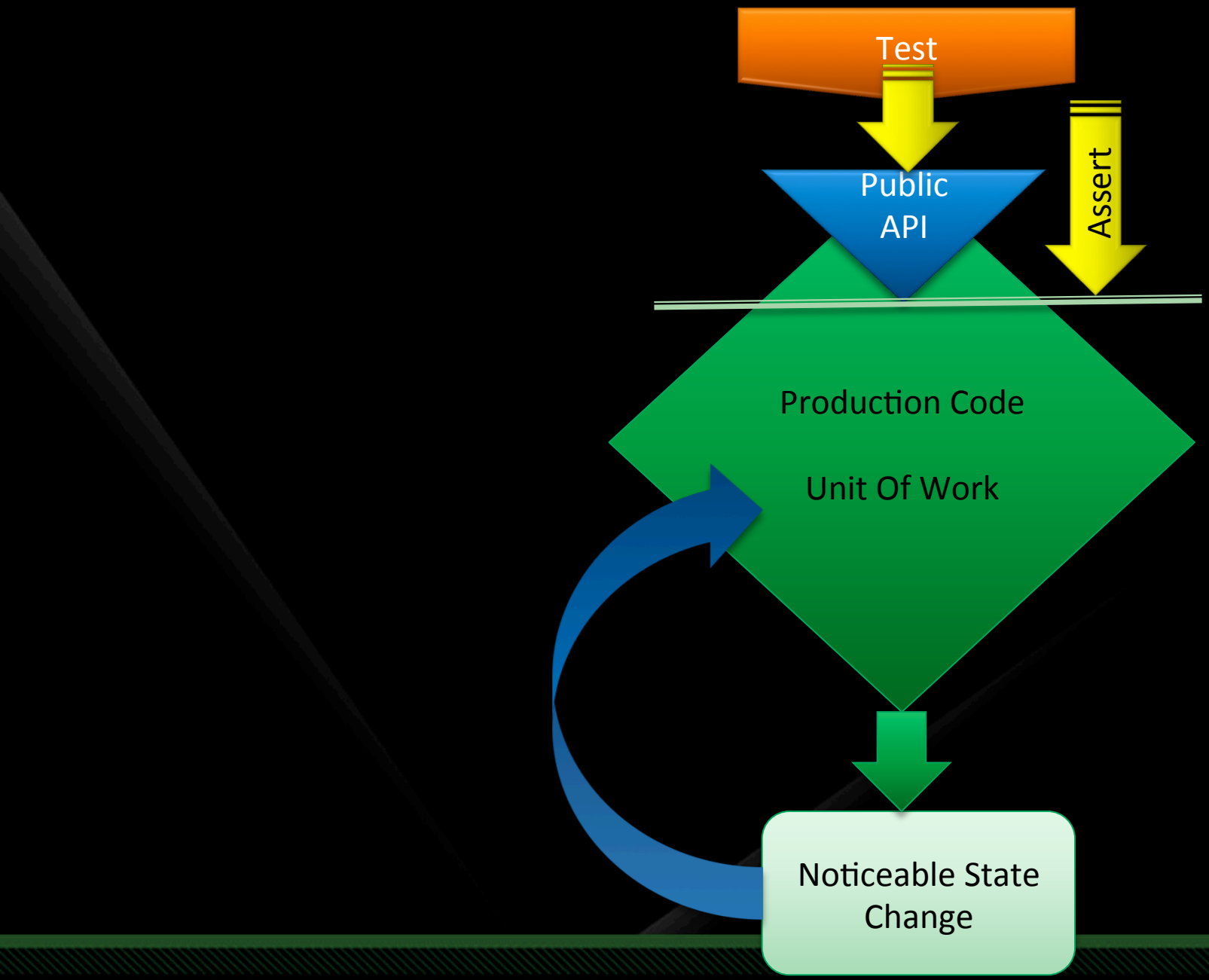
Simplified

- **New**
- **Fake** created
- If you verify calls against it – it's a mock.
- Otherwise – a stub

"Xunit Test Patterns"

- Old
- *Test Spy*
- *Test Double*
- *Fake*
- *Mock*
- *Stub*

State Based: Specifying Implementation instead of result


Test
Public
API
Assert
Production Code
Unit Of Work
Noticeable State
Change

```
test( "adds user in memory", function() {

    var userMgr = makeUserMgr();
    userMgr.addUser("user","pass");

    equal(userMgr._internalUsers[0].name,"user")
    equal(userMgr._internalUsers[0].pass,"pass")
});
```

```
test( "adds user in memory", function() {

    var userMgr = makeUserMgr();

    userMgr.addUser("user","pass");

    ok(userMgr.loginUser("user","pass"));
});
```

Mixing Unit And Integration Tests

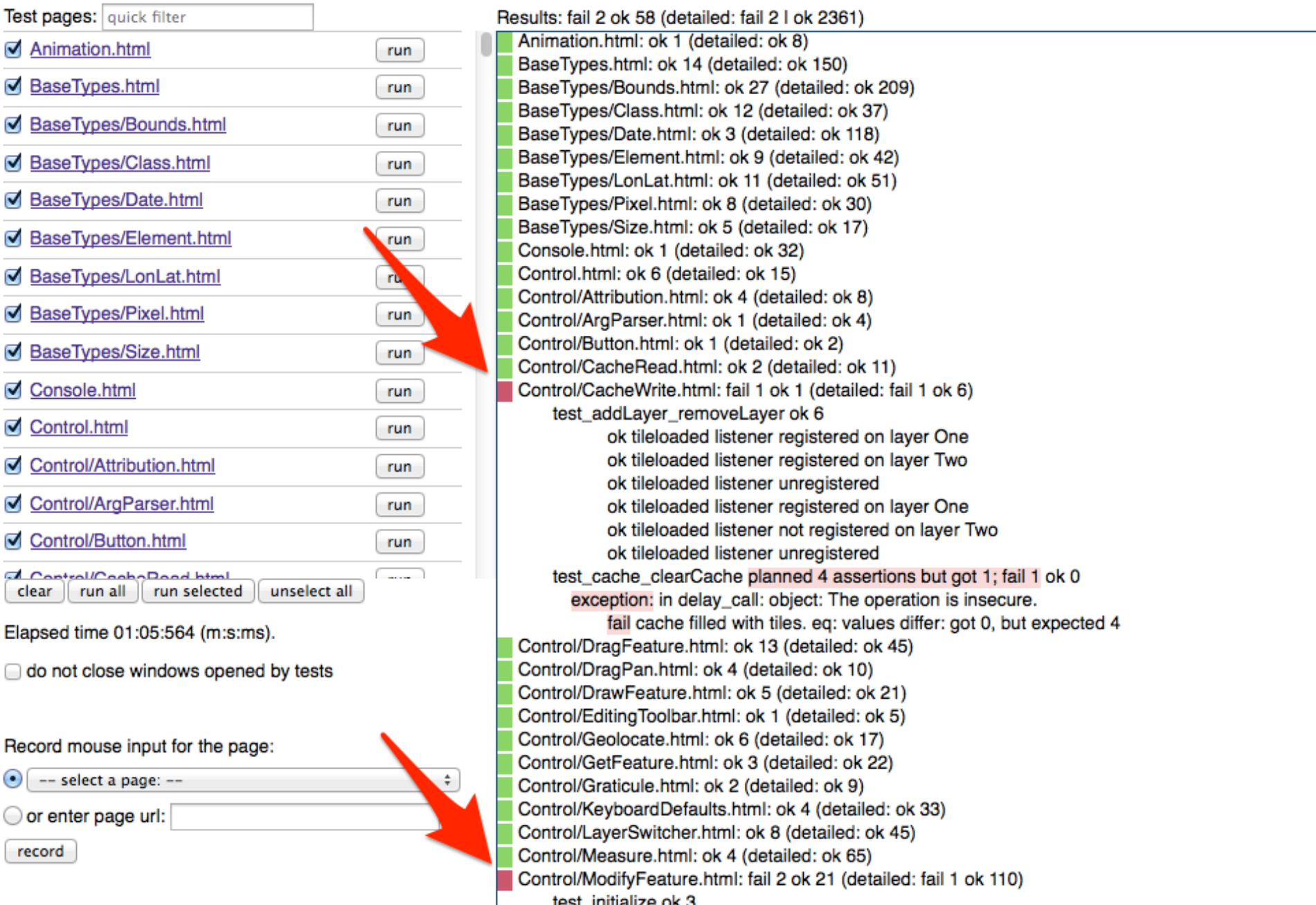
Lightweight Tests

- Repeatable
- Fast
- Consistent
- Easy to write and read

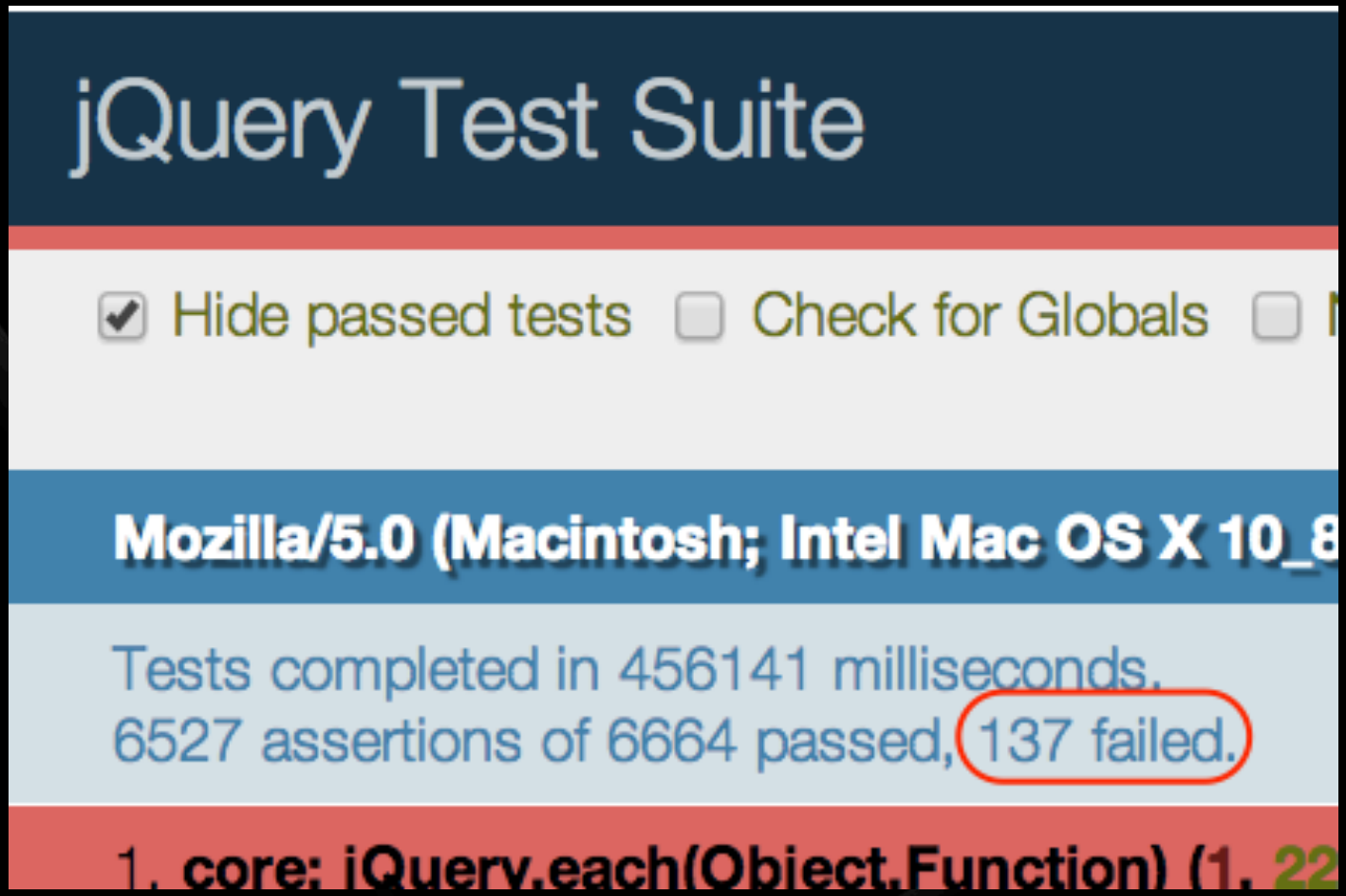
- Needs:
  - Full control of all dependencies
  - To be in Memory

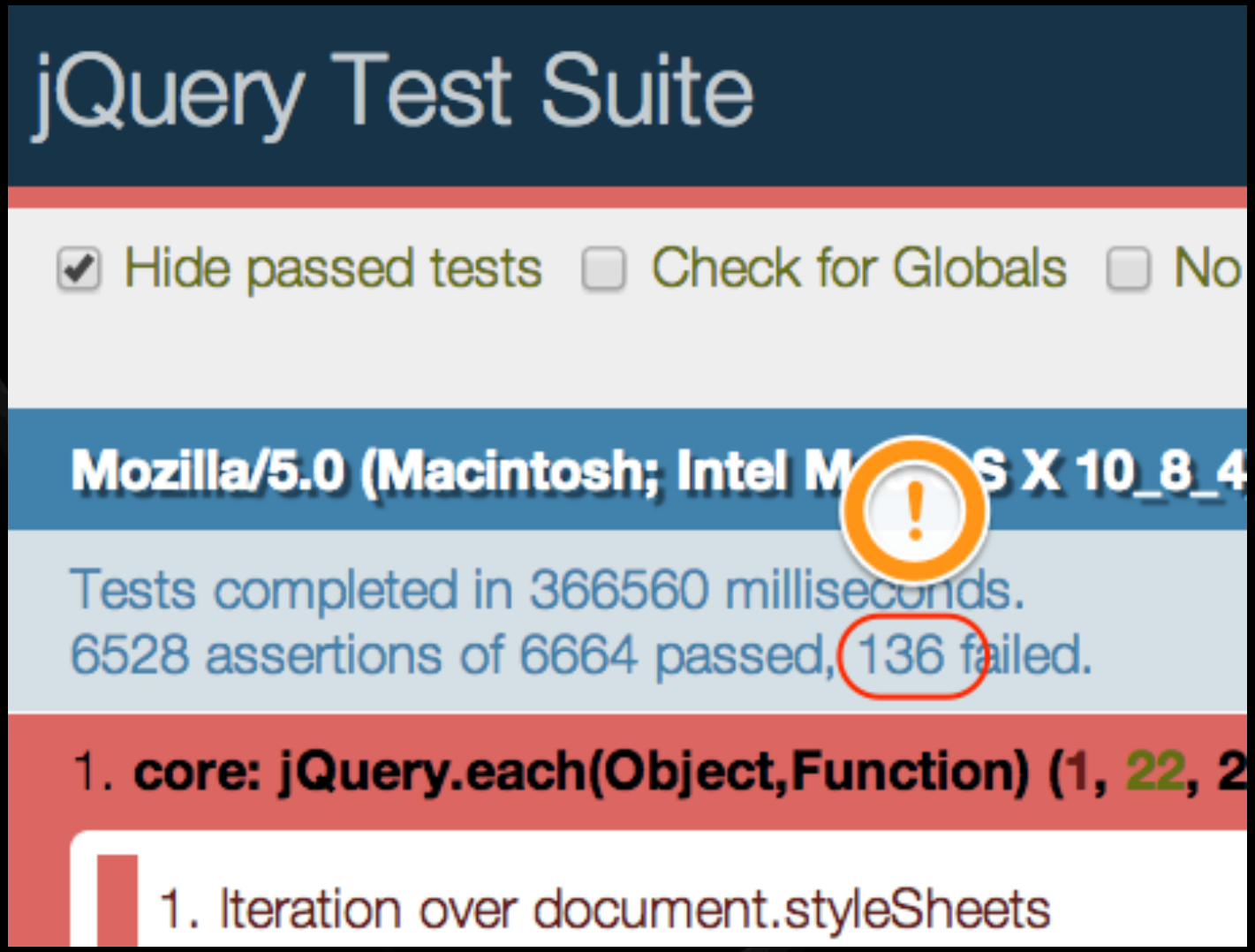
OpenLayer

Open Layer: Mixed Unit & Integration



jQUery


jQuery Test Suite
Hide passed tests
Check for Globals
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8
Tests completed in 456141 milliseconds.
6527 assertions of 6664 passed, 137 failed.
1. core: jQuery.each(Object,Function) (1, 22


jQuery Test Suite
Hide passed tests
Check for Globals
No
Mozilla/5.0 (Macintosh; Intel M
S X 10_8_4
Tests completed in 366560 milliseconds.
6528 assertions of 6664 passed, 136 failed.
1. core: jQuery.each(Object,Function) (1, 22, 2
1. Iteration over document.styleSheets

6. **ajax: jQuery.ajax() - headers (3, 1, 4)** Rerun

1. Headers were sent

Expected:

```
"siMPle: value
SometHing-elsE: other value
OthEr: something else
ajax-send: test
"
```

Result:

```
"<?php

header( \"Sample-Header: Hello World\" );
header( \"Empty-Header: \" );
header( \"Sample-Header2: Hello World 2\" );

$headers = array();

foreach( $_SERVER as $key => $value ) {

        $key = str_replace( \"_\" , \"-\" , substr( $key , 0
);
        $headers[ $key ] = $value;

}

foreach( explode( \"_\" , $_GET[ \"keys\" ] ) as $key ) {
        echo \"$key: \" . @$headers[ strtoupper( $key ) ] . \
}
"
```

Diff:

```
"siMPle: value
SometHing-elsE: other value
OthEr: something else
ajax-send: test
```

Logic in Tests

```
test("jQuery('massive html #7990')", function() {
    expect( 3 );

    var i,
        li = "<li>very very very very large html string</li>",
        html = ["<ul>"];

    for ( i = 0; i < 30000; i += 1 ) {
        html[html.length] = li;
    }
    html[html.length] = "</ul>";
    html = jQuery(html.join(""))[0];
    equal( html.nodeName.toLowerCase(), "ul");
    equal( html.firstChild.nodeName.toLowerCase(), "li");
    equal( html.childNodes.length, 30000 );
});
```

```
equal( parseInt(child.css("fontSize"), 10), 16, "Verify
equal( parseInt(child.css("font-size"), 10), 16, "Verif

child.css("height", "100%");
equal( child[0].style.height, "100%", "Make sure the he

child.attr("class", "em");
equal( parseInt(child.css("fontSize"), 10), 32, "Verify

// Have to verify this as the result depends upon the b
// support for font-size percentages
child.attr("class", "prct");
prctval = parseInt(child.css("fontSize"), 10);
checkval = 0;
if ( prctval === 16 || prctval === 24 ) {
    checkval = prctval;
}

equal( prctval, checkval, "Verify fontSize % set." );

equal( typeof child.css("width"), "string", "Make sure

old = child[0].style.height;
```

Forcing Assertion Counting

```
test( "text()", function() {

    expect( 5 );

    var expected, frag, $newLineTest;

    expected = "This link has class=\"blog\": Simon Willison's Weblog";
    equal( jQuery("#sap").text(), expected, "Check for merged text of more ther

    // Check serialization of text values
    equal( jQuery(document.createTextNode("foo")).text(), "foo", "Text node was
    notEqual( jQuery(document).text(), "", "Retrieving text for the document re

    // Retrieve from document fragments #10864
    frag = document.createDocumentFragment();
    frag.appendChild( document.createTextNode("foo") );

    equal( jQuery(frag).text(), "foo", "Document Fragment Text node was retriev

    $newLineTest = jQuery("<div>test<br/>testy</div>").appendTo("#moretests");
    $newLineTest.find("br").replaceWith("\n");
    equal( $newLineTest.text(), "test\ntesty", "text() does not remove new line

    $newLineTest.remove();
});
```

Maintainability Problem

```
test( "text()", function() {

    expect( 5 );

    var expected, frag, $newLineTest;

    expected = "This link has class=\"blog\": Simon Willison's Weblog";
    equal( jQuery("#sap").text(), expected, "Check for merged text of more ther

    // Check serialization of text values
    equal( jQuery(document.createTextNode("foo")).text(), "foo", "Text node was
    notEqual( jQuery(document).text(), "", "Retrieving text for the document re

    // Retrieve from document fragments #10864
    frag = document.createDocumentFragment();
    frag.appendChild( document.createTextNode("foo") );

    equal( jQuery(frag).text(), "foo", "Document Fragment Text node was retriev

    $newLineTest = jQuery("<div>test<br/>testy</div>").appendTo("#moretests");
    $newLineTest.find("br").replaceWith("\n");
    equal( $newLineTest.text(), "test\ntesty", "text() does not remove new line
```

John Resig @jeresig 1d

@RoyOsherove It's a bit of extra work - but it's good to know if the suite ever stops running for some weird reason.

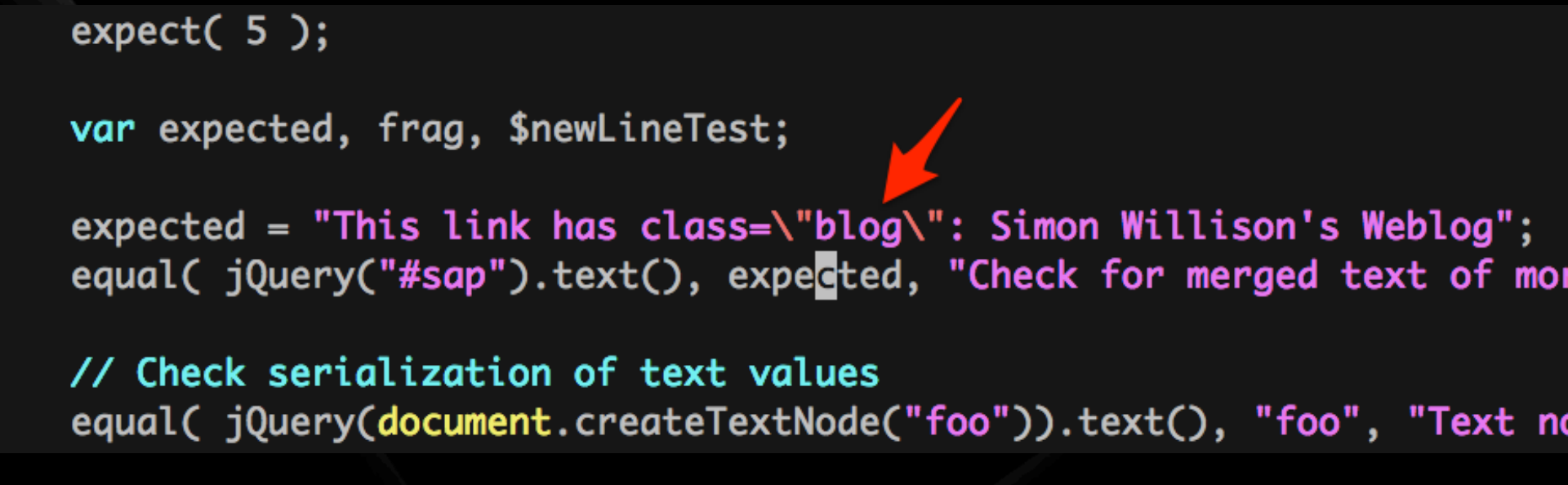
```
function testAppend( valueObj ) {

    expect( 78 );

    testAppendForObject( valueObj, false );
    testAppendForObject( valueObj, true );

    var defaultText, result, message, iframe, iframeDoc,
        $input, $radioChecked, $radioUnchecked, $radioPar

    defaultText = "Try them out:";
    result = jQuery("#first").append( valueObj("<b>buga</

    equal( result.text(), defaultText + "buga", "Check if
    equal(
        jQuery("#select3").append( valueObj("<option valu
        .find("option:last-child").attr("value"),
        "appendTest",
        "Appending html options to select element" );
```

```
test( "jQuery.when - joined", function() {

    expect( 119 );

    var deferreds = {
            value: 1,
            success: jQuery.Deferred().resolve( 1 ),
            error: jQuery.Deferred().reject( 0 ),
            futureSuccess: jQuery.Deferred().notify( true ),
            futureError: jQuery.Deferred().notify( true ),
            notify: jQuery.Deferred().notify( true )
        },
        willSucceed = {
```

```
test( "globalEval with 'use strict'", function() {
    expect( 1 );
    Globals.register("strictEvalTest");

    jQuery.globalEval("'use strict'; var strictEval
    equal( window.strictEvalTest, 1, "Test variable
});
```

Separating Test From Tested

Non obvious



```
expect( 5 );

var expected, frag, $newLineTest;

expected = "This link has class=\"blog\": Simon Willison's Weblog";
equal( jQuery("#sap").text(), expected, "Check for merged text of mo

// Check serialization of text values
equal( jQuery(document.createTextNode("foo")).text(), "foo", "Text n
```

Here it is

```
    This link has <code><a href="http://smin" id="anchor1">class="blog"</a></
    <a href="http://diveintomark.org/" class="blog" hreflang="en" id="mark">d

</p>
<div id="foo">
    <p id="sndp">Everything inside the red border is inside a div with <code>
    <p lang="en"  d="en">This is a normal link: <a id="yahoo" href="http://ww
    <p id="sap">This link has <code><a href="#2" id="anchor2">class="blog"</a

</div>
```

Seperating vars from usage

```
test( "css(String|Hash)", function() {
    expect( 40 );

    equal( jQuery("#qunit-fixture").css("display"), "block", "Check for css property \"display\"" );

    var $child, div, div2, width, height, child, prctval, checkval, old;

    $child = jQuery("#nothiddendivchild").css({ "width": "20%", "height": "20%" });
    notEqual( $child.css("width"), "20px", "Retrieving a width percentage on the child of a hidden div returns percentage" );
    notEqual( $child.css("height"), "20px", "Retrieving a height percentage on the child of a hidden div returns percentage" );

    div = jQuery( "<div>" );

    // These should be "auto" (or some better value)
    // temporarily provide "0px" for backwards compat
    equal( div.css("width"), "0px", "Width on disconnected node." );
    equal( div.css("height"), "0px", "Height on disconnected node." );

    div.css({ "width": 4, "height": 4 });

    equal( div.css("width"), "4px", "Width on disconnected node." );
    equal( div.css("height"), "4px", "Height on disconnected node." );

    div2 = jQuery( "<div style='display:none;'><input type='text' style='height:20px;'/><textarea style='height:20px;'/><div style='heigh

    equal( div2.find("input").css("height"), "20px", "Height on hidden input." );
    equal( div2.find("textarea").css("height"), "20px", "Height on hidden textarea." );
    equal( div2.find("div").css("height"), "20px", "Height on hidden textarea." );

    div2.remove();

    // handle negative numbers by setting to zero #11604
    jQuery("#nothiddendiv").css( {"width": 1, "height": 1} );

    width = parseFloat(jQuery("#nothiddendiv").css("width"));
    height = parseFloat(jQuery("#nothiddendiv").css("height"));
    jQuery("#nothiddendiv").css({ "overFlow":"hidden", "width": -1, "height": -1 });
    equal( parseFloat(jQuery("#nothiddendiv").css("width")), 0, "Test negative width set to 0");
    equal( parseFloat(jQuery("#nothiddendiv").css("height")), 0, "Test negative height set to 0");

    equal( jQuery("<div style='display: none;'>").css("display"), "none", "Styles on disconnected nodes");

    jQuery("#floatTest").css({"float": "right"});
    equal( jQuery("#floatTest").css("float"), "right", "Modified CSS float using \"float\": Assert float is right");
    jQuery("#floatTest").css({"font-size": "30px"});
    equal( jQuery("#floatTest").css("font-size"), "30px", "Modified CSS font-size: Assert font-size is 30px");
    jQuery.each("0,0.25,0.5,0.75,1".split(","), function(i, n) {
        jQuery("#foo").css({"opacity": n});

        equal( jQuery("#foo").css("opacity"), parseFloat(n), "Assert opacity is " + parseFloat(n) + " as a String" );
        jQuery("#foo").css({"opacity": parseFloat(n)});
        equal( jQuery("#foo").css("opacity"), parseFloat(n), "Assert opacity is " + parseFloat(n) + " as a Number" );
    });
    jQuery("#foo").css({"opacity": ""});
    equal( jQuery("#foo").css("opacity"), "1", "Assert opacity is 1 when set to an empty String" );

    equal( jQuery("#empty").css("opacity"), "0", "Assert opacity is accessible via filter property set in stylesheet in IE" );
    jQuery("#empty").css({ "opacity": "1" });
    equal( jQuery("#empty").css("opacity"), "1", "Assert opacity is taken from style attribute when set vs stylesheet in IE with filters"

    div = jQuery("#nothiddendiv");
    child = jQuery("#nothiddendivchild");

    equal( parseInt(div.css("fontSize"), 10), 16, "Verify fontSize px set." );
    equal( parseInt(div.css("font-size"), 10), 16, "Verify fontSize px set." );
    equal( parseInt(child.css("fontSize"), 10), 16, "Verify fontSize px set." );
    equal( parseInt(child.css("font-size"), 10), 16, "Verify fontSize px set." );

    child.css("height", "100%");
    equal( child[0].style.height, "100%", "Make sure the height is being set correctly." );

    child.attr("class", "em");
    equal( parseInt(child.css("fontSize"), 10), 32, "Verify fontSize em set." );

    // Have to verify this, the result depends upon the browser's CSS
    // support for font-size percentages
    child.attr("class", "pct");
    prctval = parseInt(child.css("fontSize"), 10);
    checkval = 0;
    if ( prctval === 16 || prctval === 24 ) {
        checkval = prctval;
    }

    equal( prctval, checkval, "Verify fontSize % set." );

    equal( typeof child.css("width"), "string", "Make sure that a string width is returned from css('width')." );

    old = child[0].style.height;
```

Comments

```
    j = jQuery( "<span>hi</span> there <!-- mon
    ok( j.length >= 2, "Check node,textnode,co

    ok( !jQuery("<option>test</option>")[0].se

    ok( jQuery("<div></div>")[0], "Create a di
    ok( jQuery("<table></table>")[0], "Create

    // equal( jQuery("element[attribute='<div>
    // equal( jQuery("element[attribute=<div></
    // equal( jQuery("element:not(<div></div>)
    // equal( jQuery("\\<div\\>").length, 0, "
);
```

Naming

Bad Name

```
test( "text(undefined)", function() {

    expect( 1 );

    equal( jQuery("#foo").text("<div").text(undefined)[ 0 ].innerHTML, "&lt;div",
    ".text(undefined) is chainable (#5571)" );
});
```

Unit Of Work
Scenario
Expected Behavior

```
test( "text(undefined)", function() {

    expect( 1 );

    equal( jQuery("#foo").text("<div").text(undefined)[ 0 ].innerHTML, "&lt;div",
    ".text(undefined) is chainable (#5571)" );
});
```

Text
with undefined
can be chained? Should not?

```
test( "text(undefined)", function() {

    expect( 1 );

    equal( jQuery("#foo").text("<div").text(undefined)[ 0 ].innerHTML, "&lt;div",
    ".text(undefined) is chainable (#5571)" );
});
```

Test of Doom

```
function testAppend( valueObj ) {

    expect( 78 );

    testAppendForObject( valueObj, false );
    testAppendForObject( valueObj, true );

    var defaultText, result, message, iframe, iframeDoc, j, d,
        $input, $radioChecked, $radioUnchecked, $radioParent, $map, $table;

    defaultText = "Try them out:";
    result = jQuery("#first").append( valueObj("<b>buga</b>") );

    equal( result.text(), defaultText + "buga", "Check if text appending works" );
    equal(
        jQuery("#select3").append( valueObj("<option value='appendTest'>Append Test</option>") )
        .find("option:last-child").attr("value"),
        "appendTest",
        "Appending html options to select element" );

    jQuery("form").append( valueObj("<input name='radiotest' type='radio' checked='checked' />") );
    jQuery("form input[name=radiotest]").each(function() {
        ok( jQuery(this).is(":checked"), "Append checked radio" );
    }).remove();

    jQuery("form").append( valueObj("<input name='radiotest2' type='radio' checked   =   'checked' />") );
    jQuery("form input[name=radiotest2]").each(function() {
        ok( jQuery(this).is(":checked"), "Append alternately formated checked radio" );
    }).remove();

    jQuery("form").append( valueObj("<input name='radiotest3' type='radio' checked />") );
    jQuery("form input[name=radiotest3]").each(function() {
        ok( jQuery(this).is(":checked"), "Append HTML5-formated checked radio" );
    }).remove();

    jQuery("form").append( valueObj("<input type='radio' checked='checked' name='radiotest4' />") );
    jQuery("form input[name=radiotest4]").each(function() {
        ok( jQuery(this).is(":checked"), "Append with name attribute after checked attribute" );
    }).remove();

    message = "Test for appending a DOM node to the contents of an iframe";
    iframe = jQuery("#iframe")[ 0 ];
    iframeDoc = iframe.contentDocument || iframe.contentWindow && iframe.contentWindow.document;

    try {
        if ( iframeDoc && iframeDoc.body ) {
            equal( jQuery(iframeDoc.body).append( valueObj("<div id='success'>test</div>") )[ 0 ].lastChild.id, "success", message );
        } else {
            ok( true, message + " - can't test" );
        }
    } catch( e ) {
        strictEqual( e.message || e, undefined, message );
    }

    jQuery("<fieldset/>").appendTo("#form").append( valueObj("<legend id='legend'>test</legend>") );
    t( "Append legend", "#legend", [ "legend" ] );

    $map = jQuery("<map/>").append( valueObj("<area id='map01' shape='rect' coords='50,50,150,150' href='http://www.jquery.com/' alt='jQuery'>") );

    equal( $map[ 0 ].childNodes.length, 1, "The area was inserted." );
    equal( $map[ 0 ].firstChild.nodeName.toLowerCase(), "area", "The area was inserted." );

    jQuery("#select1").append( valueObj("<OPTION>Test</OPTION>") );
    equal( jQuery("#select1 option:last-child").text(), "Test", "Appending OPTION (all caps)" );

    jQuery("#select1").append( valueObj("<optgroup label='optgroup'><option>optgroup</option></optgroup>") );
    equal( jQuery("#select1 optgroup").attr("label"), "optgroup", "Label attribute in newly inserted optgroup is correct" );
    equal( jQuery("#select1 option").last().text(), "optgroup", "Appending optgroup" );

    $table = jQuery("#table");

    jQuery.each( "thead tbody tfoot colgroup caption tr th td".split(" "), function( i, name ) {
        $table.append( valueObj( "<" + name + "/>" ) );
        equal( $table.find( name ).length, 1, "Append " + name );
        ok( jQuery.parseHTML( "<" + name + "/>" ).length, name + " wrapped correctly" );
    });

    jQuery("#table colgroup").append( valueObj("<col/>") );
    equal( jQuery("#table colgroup col").length, 1, "Append col" );

    jQuery("#form")
        .append( valueObj("<select id='appendSelect1'></select>") )
        .append( valueObj("<select id='appendSelect2'><option>Test</option></select>") );
    t( "Append Select", "#appendSelect1, #appendSelect2", [ "appendSelect1", "appendSelect2" ] );

    equal( "Two nodes", jQuery("<div />").append( "Two", " nodes" ).text(), "Appending two text nodes (#4011)" );
    equal( jQuery("<div />").append( "1", "", 3 ).text(), "13", "If median is falsy-like value, subsequent arguments should not be ignored" );

    // using contents will get comments regular, text, and comment nodes
    j = jQuery("#nonnodes").contents();
    d = jQuery("<div/>").appendTo("#nonnodes").append( j );

    equal( jQuery("#nonnodes").length, 1, "Check node,textnode,comment append moved leaving just the div" );
    equal( d.contents().length, 3, "Check node,textnode,comment append works" );
    d.contents().appendTo("#nonnodes");
    d.remove();
    equal( jQuery("#nonnodes").contents().length, 3, "Check node,textnode,comment append cleanup worked" );

    $input = jQuery("<input type='checkbox'/>").prop( "checked", true ).appendTo("#testForm");
    equal( $input[ 0 ].checked, true, "A checked checkbox that is appended stays checked" );

    $radioChecked = jQuery("input[type='radio'][name='R1']").eq( 1 );
    $radioParent = $radioChecked.parent();
    $radioUnchecked = jQuery("<input type='radio' name='R1' checked='checked'/>").appendTo( $radioParent );
    $radioChecked.trigger("click");
    $radioUnchecked[ 0 ].checked = false;

    jQuery("<div/>").insertBefore($radioParent).append($radioParent);

    equal( $radioChecked[ 0 ].checked, true, "Reappending radios uphold which radio is checked" );
    equal( $radioUnchecked[ 0 ].checked, false, "Reappending radios uphold not being checked" );

    equal( jQuery("<div/>").append( valueObj("<area/>") )[ 0 ].childNodes.length, 2, "HTML-string with leading text should be processed correctly" );
}
```

```
function testAppend( valueObj ) {

    expect( 78 );

    testAppendForObject( valueObj, false );
    testAppendForObject( valueObj, true );

    var defaultText, result, message, iframe, iframeDoc,
        $input, $radioChecked, $radioUnchecked, $radioPar

    defaultText = "Try them out:";
    result = jQuery("#first").append( valueObj("<b>buga</

    equal( result.text(), defaultText + "buga", "Check if
    equal(
        jQuery("#select3").append( valueObj("<option valu
        .find("option:last-child").attr("value"),
        "appendTest",
        "Appending html options to select element" );
```

```
jQuery("form input[name=radiotest4]").each(function() {
    ok( jQuery(this).is(":checked"), "Append with name attribute after checked at
}).remove();

message = "Test for appending a DOM node to the contents of an iframe";
iframe = jQuery("#iframe")[ 0 ];
iframeDoc = iframe.contentDocument || iframe.contentWindow && iframe.contentWindo

try {
    if ( iframeDoc && iframeDoc.body ) {
        equal( jQuery(iframeDoc.body).append( valueObj("<div id='success'>test</d
    } else {
        ok( true, message + " - can't test" );
    }
} catch( e ) {
    strictEqual( e.message || e, undefined, message );
}

jQuery("<fieldset/>").appendTo("#form").append( valueObj("<legend id='legend'>tes
t( "Append legend", "#legend", [ "legend" ] );

$map = jQuery("<map/>").append( valueObj("<area id='map01' shape='rect' coords='5

equal( $map[ 0 ].childNodes.length, 1, "The area was inserted." );
equal( $map[ 0 ].firstChild.nodeName.toLowerCase(), "area", "The area was inserte

jQuery("#select1").append( valueObj("<OPTION>Test</OPTION>") );
equal( jQuery("#select1 option:last-child").text(), "Test", "Appending OPTION (al
```

```
message = "Test for appending a DOM node to the contents of an iframe";
iframe = jQuery("#iframe")[ 0 ];
iframeDoc = iframe.contentDocument || iframe.contentWindow && iframe.contentWindo

try {
    if ( iframeDoc && iframeDoc.body ) {
        equal( jQuery(iframeDoc.body).append( valueObj("<div id='success'>test</d
    } else {
        ok( true, message + " - can't test" );
    }
} catch( e ) {
    strictEqual( e.message || e, undefined, message );
}
```

Read Maintain

Multiple Tests in Single Test

```
test( "append(Function) with incoming value", function() {

    expect( 12 );

    var defaultText, result, select, old, expected;

    defaultText = "Try them out:";
    old = jQuery("#first").html();

    result = jQuery("#first").append(function( i, val ) {
        equal( val, old, "Make sure the incoming value is correct." );
        return "<b>buga</b>";
    });
    equal( result.text(), defaultText + "buga", "Check if text appending works" );

    select = jQuery("#select3");
    old = select.html();

    equal( select.appe(function( i, val ) {
        equal( val, o, "Make sure the incoming value is correct." );
        return "<o n value='appendTest'>Append Test</option>";
    }).find("optio st-child").attr("value"), "appendTest", "Appending html options to select element" );

    QUnit.reset();
    expected = "This link has class=\"blog\": Simon Willison's WeblogTry them out:";
    old = jQuery("#sap").html();

    jQuery("#sap").append(functi on( i, val ) {
        equal( val, old, "M e sure the incoming value is correct." );
        return document. tElementById("first");
    });
    equal( jQuery(" ").text(), expected, "Check for appending of element" );

    QUnit.reset();
    expected = "This link has class=\"blog\": Simon Willison's WeblogTry them out:Yahoo";
    old = jQuery("#sap").html();

    jQuery("#sap").append(function( i, val ) {
        equal( val, old, "Make sure the incoming value is correct." );
```

Backbone

Magic Numbers

```
test("isNew", 6, function() {
  var a = new Backbone.Model({ 'foo': 1, 'bar': 2, 'baz': 3});
  ok(a.isNew(), "it should be new");
  a = new Backbone.Model({ 'foo': 1, 'bar': 2, 'baz': 3, 'id': -5 });
  ok(!a.isNew(), "any defined ID is legal, negative or positive");
  a = new Backbone.Model({ 'foo': 1, 'bar': 2, 'baz': 3, 'id': 0 });
  ok(!a.isNew(), "any defined ID is legal, including zero");
  ok( new Backbone.Model({           }).isNew(), "is true when there is no id");
  ok(!new Backbone.Model({ 'id': 2   }).isNew(), "is false for a positive integer");
  ok(!new Backbone.Model({ 'id': -5  }).isNew(), "is false for a negative integer");
});
```

```
test("isNew", 6, function() {
  var a = new Backbone.Model({ 'foo': 1, 'bar': 2, 'baz': 3});
  ok(a.isNew(), "it should be new");
  a = new Backbone.Model({ 'foo': 1, 'bar': 2, 'baz': 3, 'id': -5 });
  ok(!a.isNew(), "any defined ID is legal, negative or positive");
  a = new Backbone.Model({ 'foo': 1, 'bar': 2, 'baz': 3, 'id': 0 });
  ok(!a.isNew(), "any defined ID is legal, including zero");
  ok( new Backbone.Model({           }).isNew(), "is true when there is no id");
  ok(!new Backbone.Model({ 'id': 2   }).isNew(), "is false for a positive integer");
  ok(!new Backbone.Model({ 'id': -5  }).isNew(), "is false for a negative integer");
});
```

Naming

Handlebars

Missing Name Info

```
describe('blocks', function() {
  it("array", function() {
    var string   = "{{#goodbyes}}{{text}}! {{/goodbyes}}cruel {{world}}!";
    var hash     = {goodbyes: [{text: "goodbye"}, {text: "Goodbye"}, {text: "GOODBYE"}],
    shouldCompileTo(string, hash, "goodbye! Goodbye! GOODBYE! cruel world!",
                    "Arrays iterate over the contents when not empty");

    shouldCompileTo(string, {goodbyes: [], world: "world"}, "cruel world!",
                    "Arrays ignore the contents when empty");

  });

  it("array with @index", function() {
    var string = "{{#goodbyes}}{{@index}}. {{text}}! {{/goodbyes}}cruel {{world}}!";
    var hash   = {goodbyes: [{text: "goodbye"}, {text: "Goodbye"}, {text: "GOODBYE"}], wo

    var template = CompilerContext.compile(string);
    var result = template(hash);

    equal(result, "0. goodbye! 1. Goodbye! 2. GOODBYE! cruel world!", "The @index variabl
  });
```

Describe “Blocks”

- Describe “Array”
  - Describe “That is non empty”
    - It “iterates over the contents”
  - Describe “that is empty”
    - It “ignores the content”
  - Describe “with @index”
    - It “uses the @index variable”

Knockout

```
describe('Binding dependencies', function() {
    beforeEach(jasmine.prepareTestNode);

    it('If the binding handler depends on an observable, invokes the init handler once and the update handler whenever a new value is
        var observable = new ko.observable();
        var initPassedValues = [], updatePassedValues = [];
        ko.bindingHandlers.test = {
            init: function (element, valueAccessor) { initPassedValues.push(valueAccessor()()); },
            update: function (element, valueAccessor) { updatePassedValues.push(valueAccessor()()); }
        };
        testNode.innerHTML = "<div data-bind='test: myObservable'></div>";

        ko.applyBindings({ myObservable: observable }, testNode);
        expect(initPassedValues.length).toEqual(1);
        expect(updatePassedValues.length).toEqual(1);
        expect(initPassedValues[0]).toBeUndefined();
        expect(updatePassedValues[0]).toBeUndefined();

        observable("A");
        expect(initPassedValues.length).toEqual(1);
        expect(updatePassedValues.length).toEqual(2);
        expect(updatePassedValues[1]).toEqual("A");
    });

    it('If the associated DOM element was removed by KO, handler subscriptions are disposed immediately', function () {
        var observable = new ko.observable("A");
        ko.bindingHandlers.anyHandler = {
            update: function (element, valueAccessor) { valueAccessor(); }
        };
        testNode.innerHTML = "<div data-bind='anyHandler: myObservable()'></div>";
```

Describe “Binding Dependency”

- Describe “When handler depends on observable and new value available”
  - It “invokes the init handler”
- Describe “when associated DOM element was removed by KO”
  - It “removes handler subscriptions immediately”

Semi-Irrelevant Setup

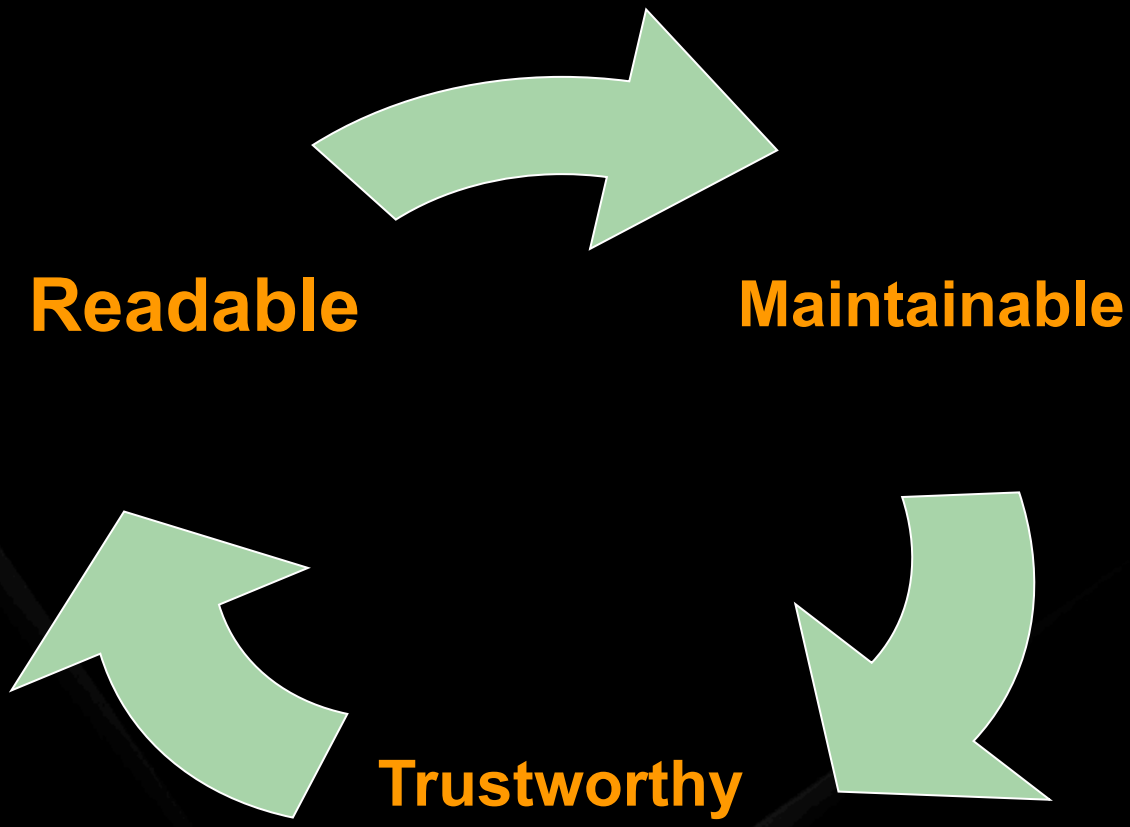
Batman

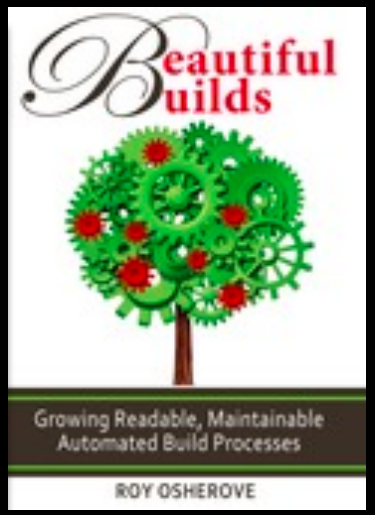
```coffee
QUnit.module "Batman.ControllerActionFrame",
  setup: ->
    @completeSpy = createSpy()
    @frame = new Batman.ControllerActionFrame({}, @completeSpy)

suite = ->
```

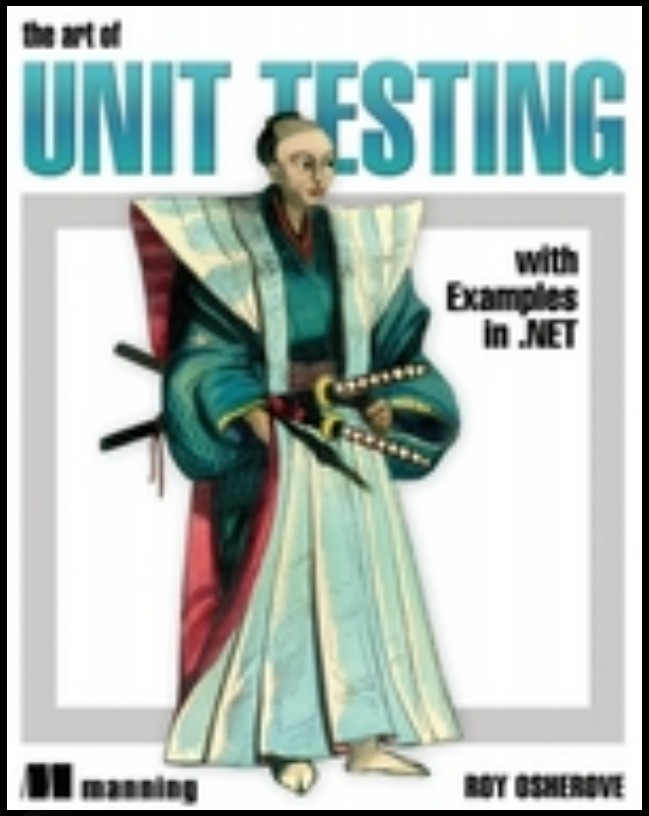
```coffee
QUnit.module "Batman.ControllerActionFrame",
  setup: ->
    @completeSpy = createSpy()
    @frame = new Batman.ControllerActionFrame({}, @completeSpy)

suite = ->
  test "starting an operation marks operationOccurred on the frame", ->
    @frame.startOperation()
    ok @frame.operationOccurred

  test "starting an operation with {internal: true} does not mark operationOccurred o
    @frame.startOperation({internal: true})
    ok !@frame.operationOccurred

  test "taking one operation fires the complete callback", ->
    @frame.startAndFinishOperation()
    ok @completeSpy.called

  test "starting one operation does not fire the complete callback", ->
    @frame.startOperation()
    ok !@completeSpy.called

  test "finishing a started operation fires the complete callback", ->
    @frame.startOperation()
    @frame.finishOperation()
    ok @completeSpy.called

  test "finishing one of many outstanding s  actions does not the complete callb
    @frame.startOperation()
    @frame.startOperation()
```

Testing Multiple Concerns

```
test 'redirecting using @redirect() in an action prevents implicit render', ->
  Batman.navigator = {redirect: createSpy()}
  @controller.test = -> @redirect '/'
  @controller.dispatch 'test'
  equal Batman.currentApp.layout.subviews.length, 0
  ok Batman.navigator.redirect.called
```


Readable
Maintainable
Trustworthy





ArtOfUnitTesting.com
@RoyOsherove