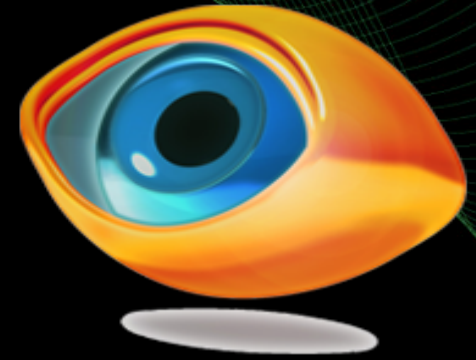


Unit Testing Best Practices and Horrible Mistakes



In JS

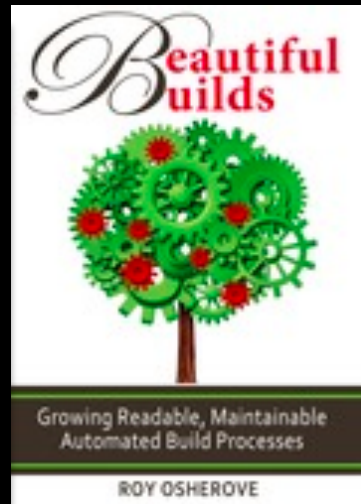
@RoyOshero

ArtOfUnitTesting.com

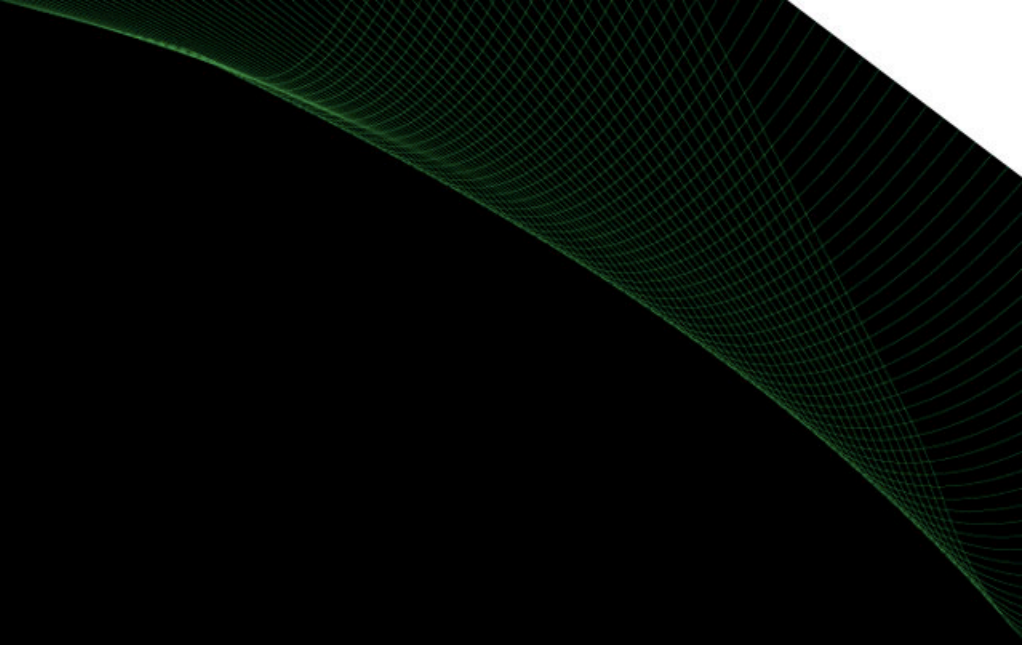
Unit Testing **Good** Practices and **Possible** Mistakes

In JS





ArtOfUnitTesting.com
@RoyOsheroove



Fail-First Experience

Possible
~~Horrible~~ Mistake

**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**

Possible
~~Horrible~~ Mistake

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

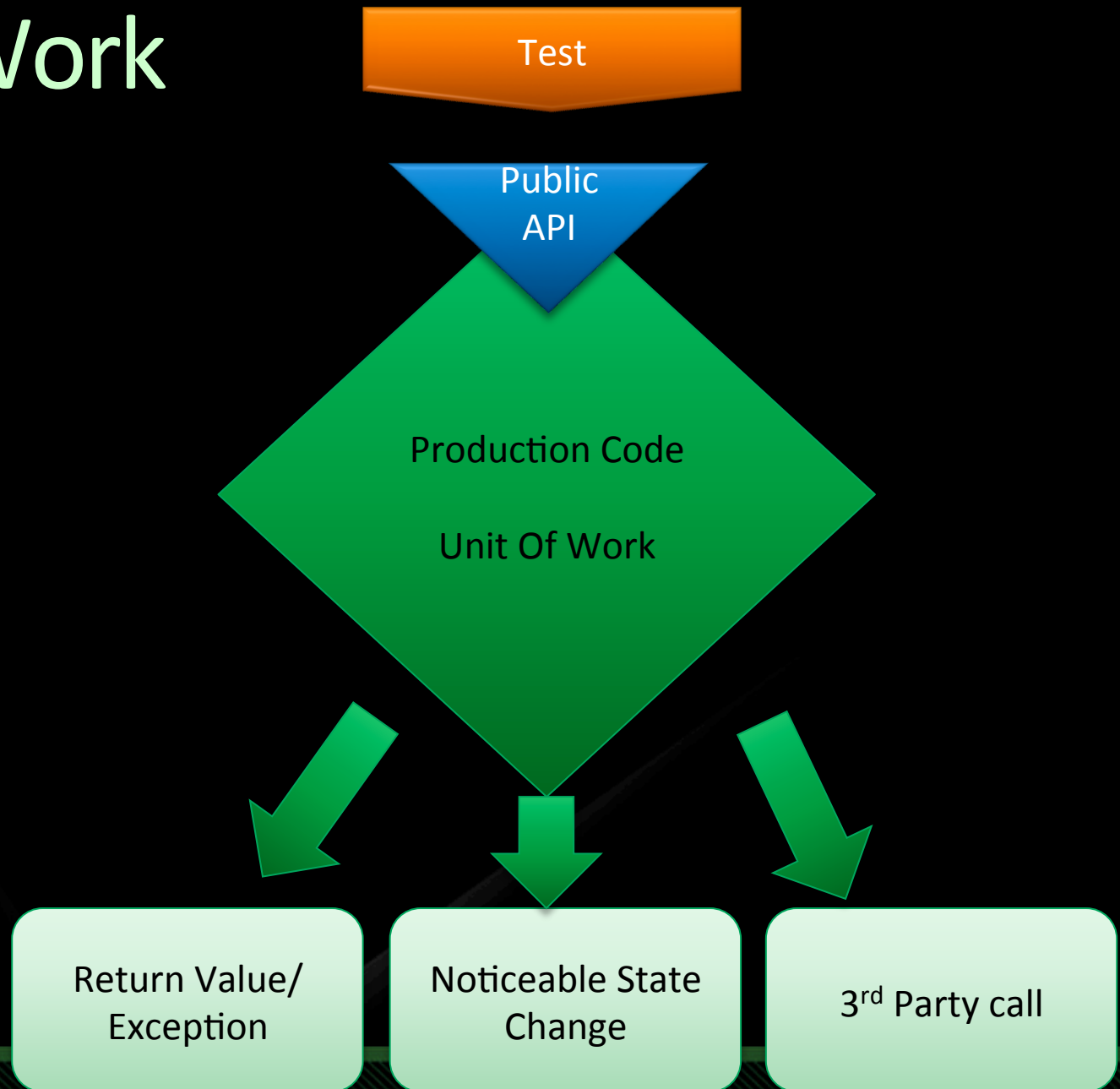
Possible
~~Horrible~~ Mistake

Faking
~~Mocking~~ all the things

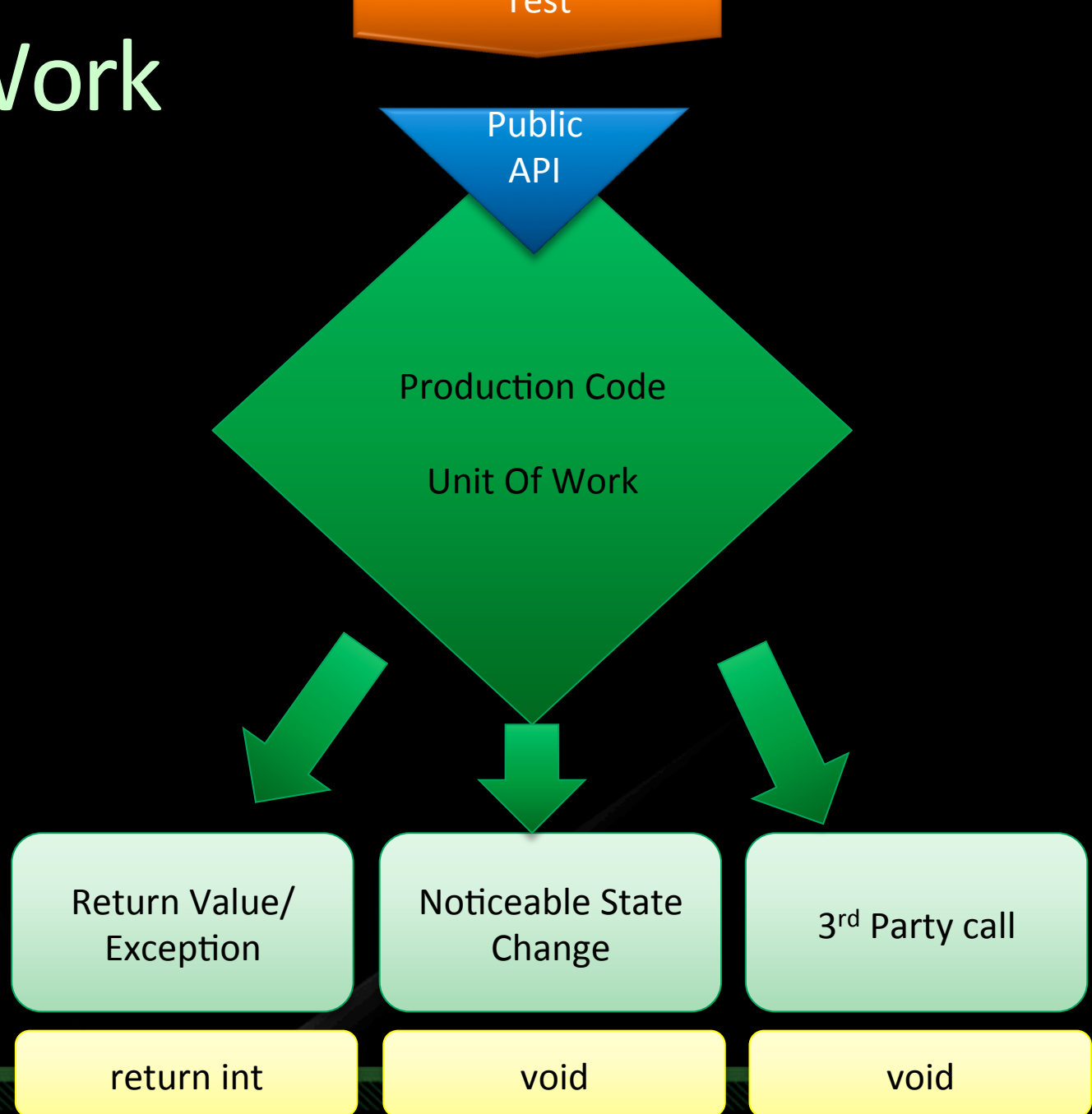
Seriously

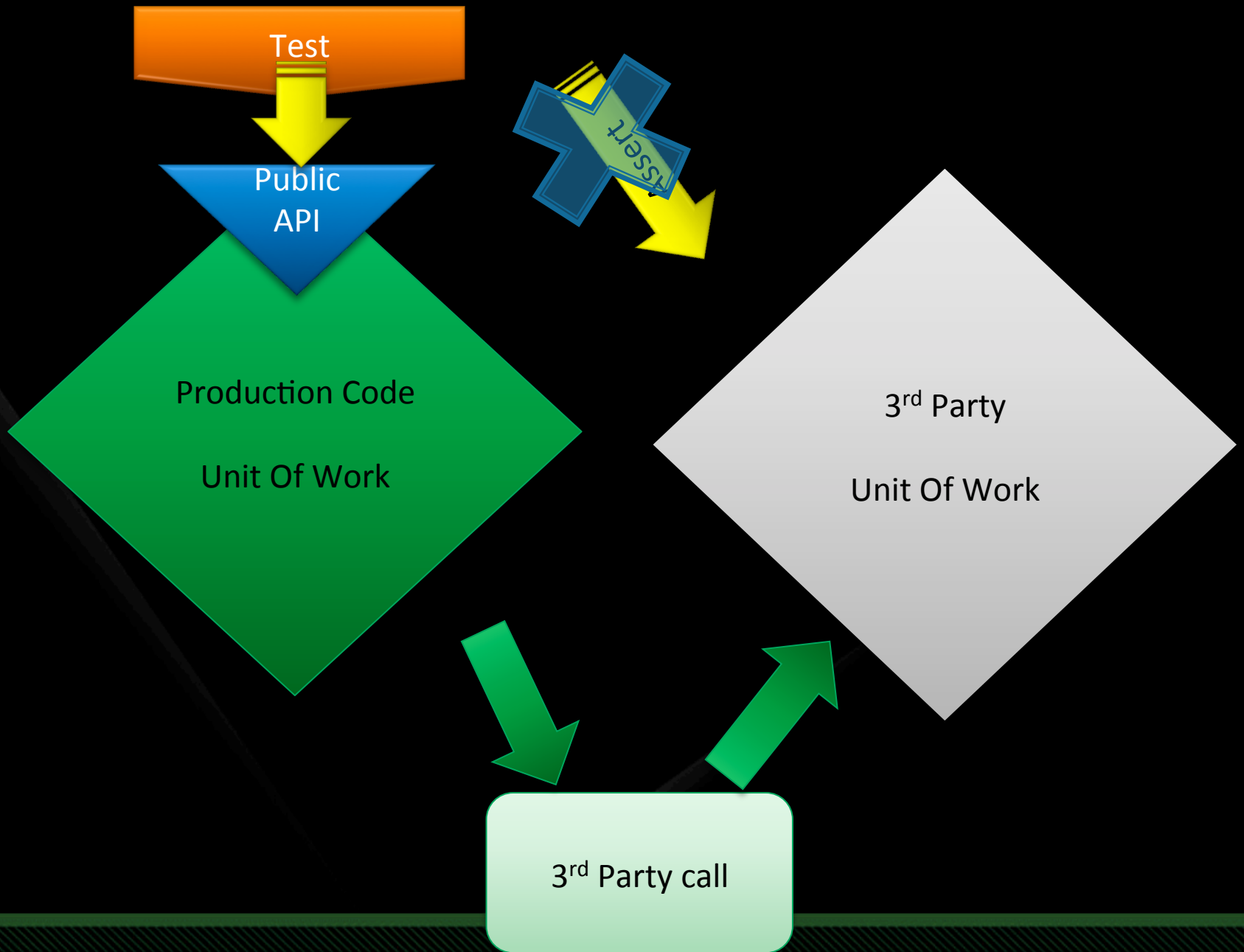
**#Tests with mock objects
.should() < 5%**

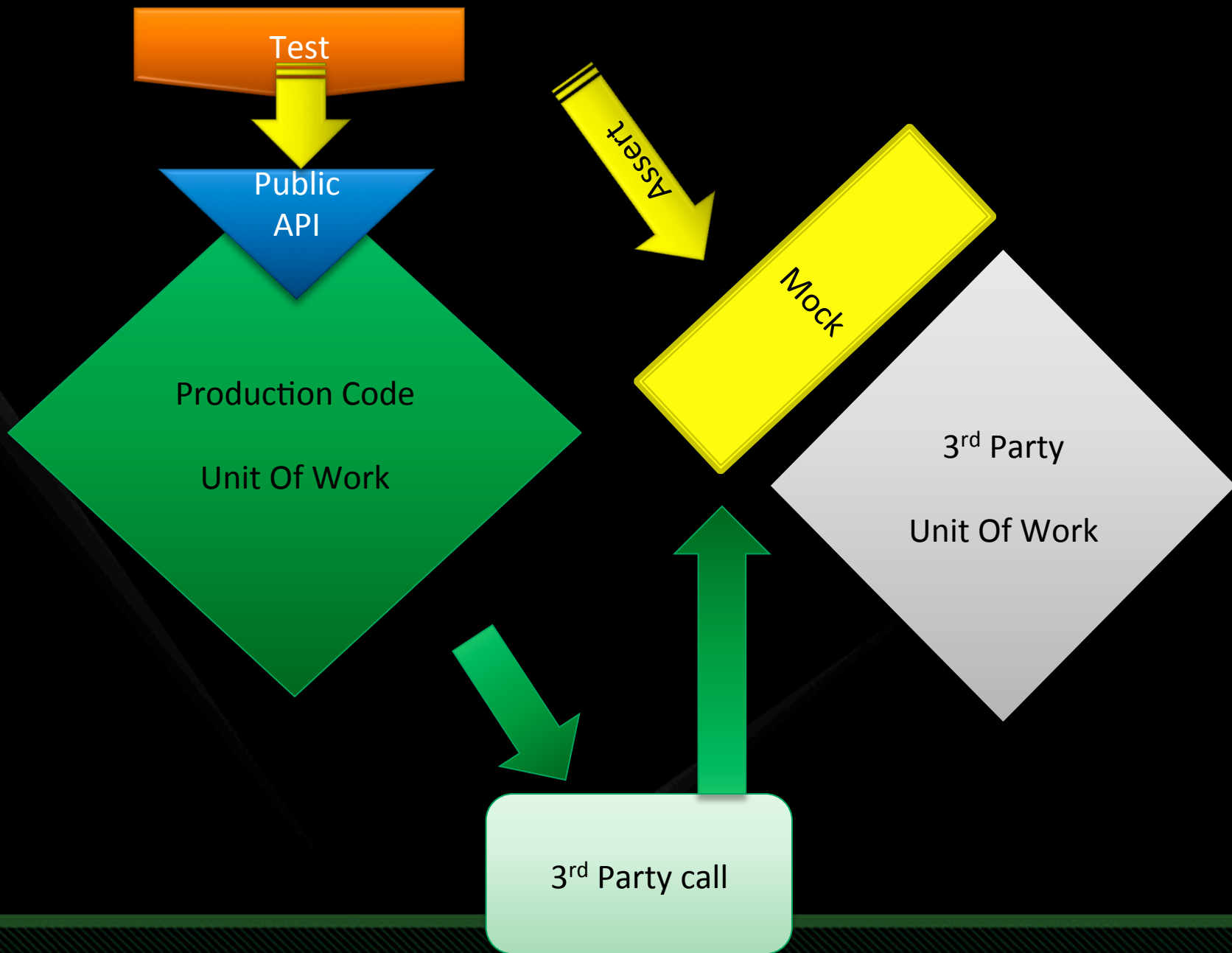
Unit Of Work



Unit Of Work







Unit Of Work



Possible
~~Horrible~~ Mistake

Everything is a
mock

A simpler terminology

“Xunit Test Patterns”

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

A simpler terminology

Simplified

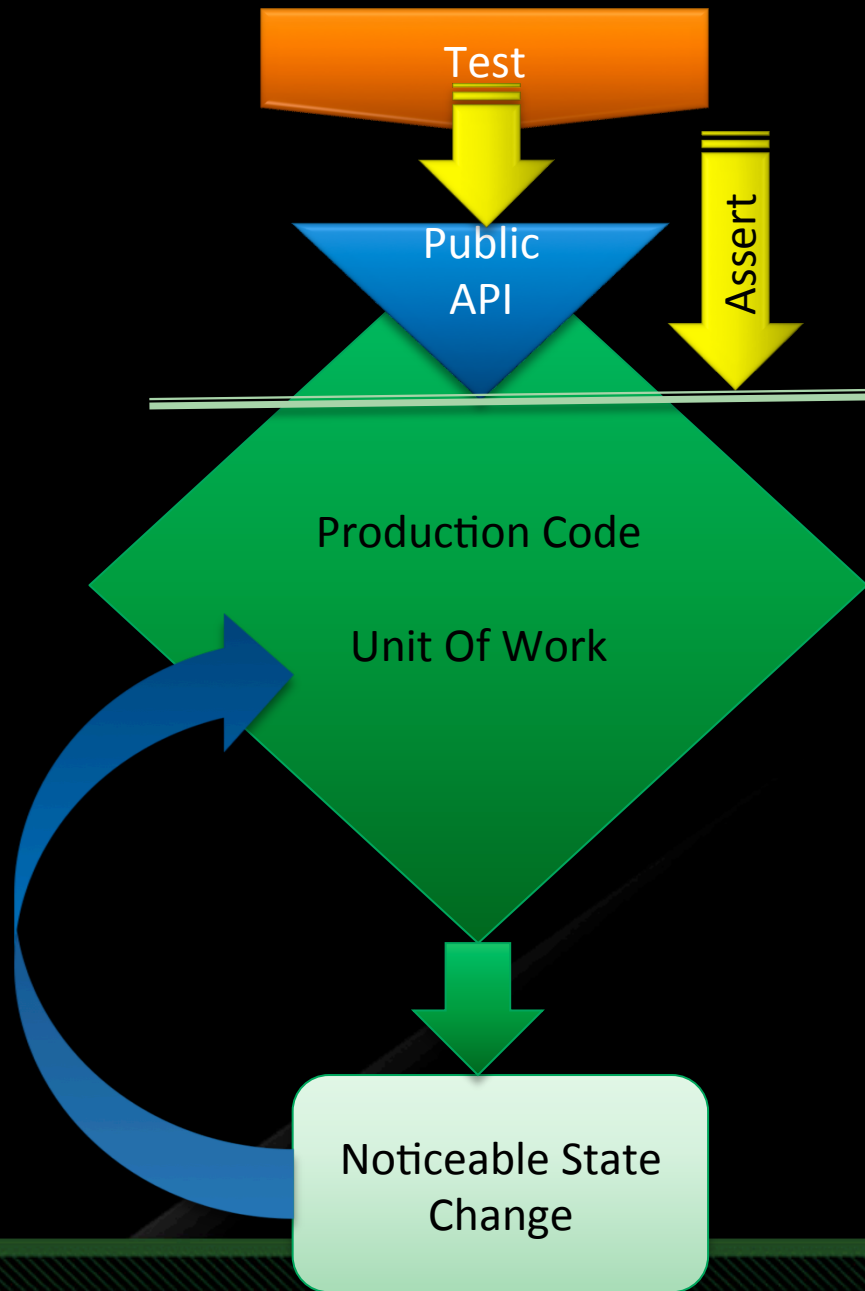
- New
- **Fake** created
- If you verify calls against it – it's a **mock**.
- Otherwise – a **stub**

“Xunit Test Patterns”

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

Possible
~~Horrible~~ Mistake

**State Based:
Specifying
Implementation
instead of result**



```
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"));  
});
```



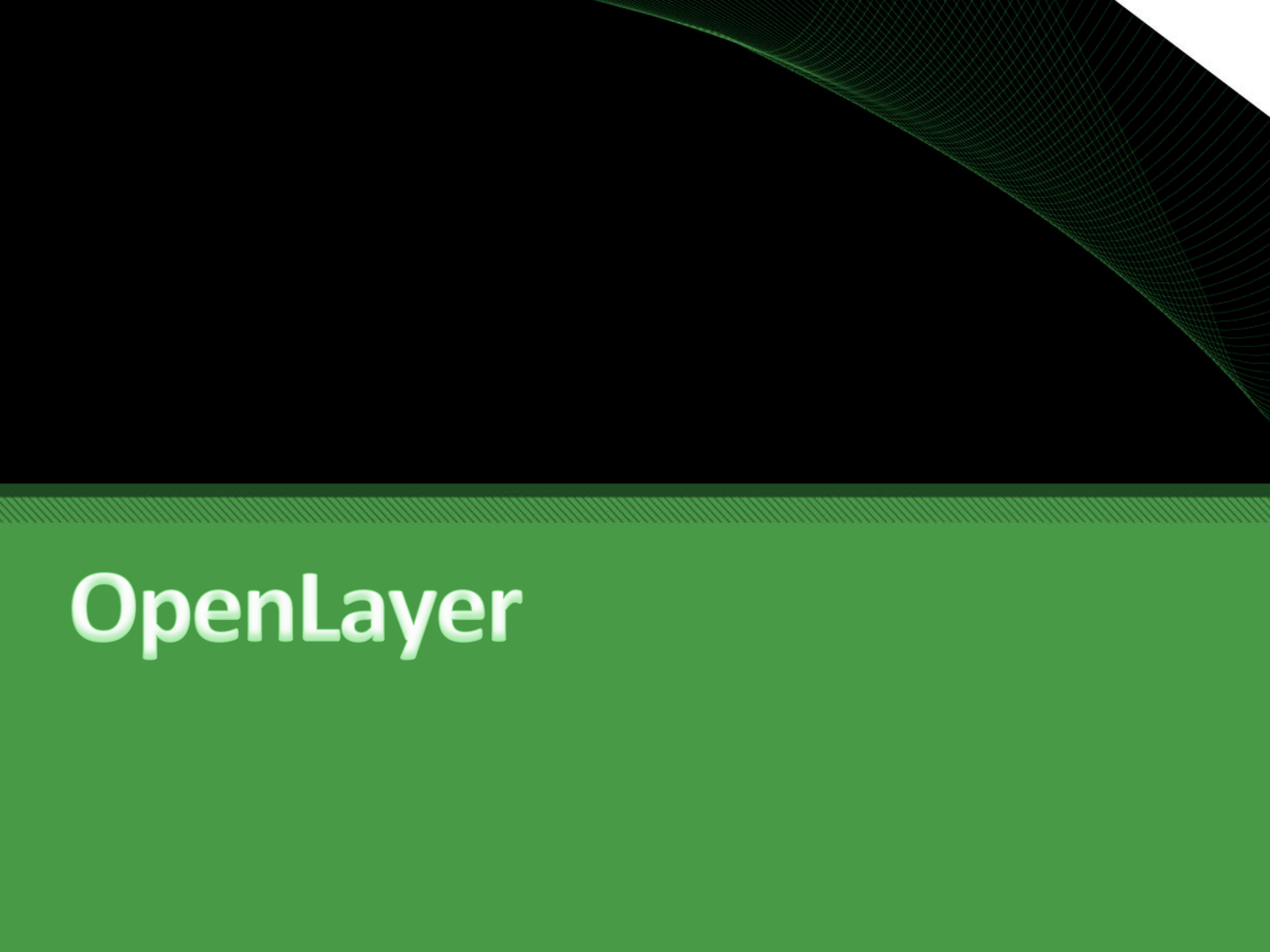


Trust

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

Test pages:

☒ Animation.html	run
☒ BaseTypes.html	run
☒ BaseTypes/Bounds.html	run
☒ BaseTypes/Class.html	run
☒ BaseTypes/Date.html	run
☒ BaseTypes/Element.html	run
☒ BaseTypes/LonLat.html	run
☒ BaseTypes/Pixel.html	run
☒ BaseTypes/Size.html	run
☒ Console.html	run
☒ Control.html	run
☒ Control/Attribution.html	run
☒ Control/ArgParser.html	run
☒ Control/Button.html	run
☒ Control/CacheRead.html	run
clear run all run selected unselect all	

Elapsed time 01:05:564 (m:s:ms).

☐ do not close windows opened by tests

Record mouse input for the page:

☒ -- select a page: --
☐ or enter page url:

Results: fail 2 ok 58 (detailed: fail 2 | ok 2361)

```
Animation.html: ok 1 (detailed: ok 8)
BaseTypes.html: ok 14 (detailed: ok 150)
BaseTypes/Bounds.html: ok 27 (detailed: ok 209)
BaseTypes/Class.html: ok 12 (detailed: ok 37)
BaseTypes/Date.html: ok 3 (detailed: ok 118)
BaseTypes/Element.html: ok 9 (detailed: ok 42)
BaseTypes/LonLat.html: ok 11 (detailed: ok 51)
BaseTypes/Pixel.html: ok 8 (detailed: ok 30)
BaseTypes/Size.html: ok 5 (detailed: ok 17)
Console.html: ok 1 (detailed: ok 32)
Control.html: ok 6 (detailed: ok 15)
Control/Attribution.html: ok 4 (detailed: ok 8)
Control/ArgParser.html: ok 1 (detailed: ok 4)
Control/Button.html: ok 1 (detailed: ok 2)
Control/CacheRead.html: ok 2 (detailed: ok 11)
Control/CacheWrite.html: fail 1 ok 1 (detailed: fail 1 ok 6)
  test_addLayer_removeLayer ok 6
    ok tileloaded listener registered on layer One
    ok tileloaded listener registered on layer Two
    ok tileloaded listener unregistered
    ok tileloaded listener registered on layer One
    ok tileloaded listener not registered on layer Two
    ok tileloaded listener unregistered
  test_cache_clearCache planned 4 assertions but got 1; fail 1 ok 0
    exception: in delay_call: object: The operation is insecure.
    fail cache filled with tiles. eq: values differ: got 0, but expected 4
Control/DragFeature.html: ok 13 (detailed: ok 45)
Control/DragPan.html: ok 4 (detailed: ok 10)
Control/DrawFeature.html: ok 5 (detailed: ok 21)
Control/EditingToolbar.html: ok 1 (detailed: ok 5)
Control/Geolocate.html: ok 6 (detailed: ok 17)
Control/GetFeature.html: ok 3 (detailed: ok 22)
Control/Graticule.html: ok 2 (detailed: ok 9)
Control/KeyboardDefaults.html: ok 4 (detailed: ok 33)
Control/LayerSwitcher.html: ok 8 (detailed: ok 45)
Control/Measure.html: ok 4 (detailed: ok 65)
Control/ModifyFeature.html: fail 2 ok 21 (detailed: fail 1 ok 110)
  test_initialize ok 3
```

The image features the jQuery logo, which consists of the word "jQuery" in a white, sans-serif font. The logo is positioned on a solid green background. Above the green background, there is a black area with a green, curved, grid-like pattern that resembles a wireframe or a stylized arc. A thin, horizontal green line with a diagonal hatching pattern separates the black area from the green background.

jQuery

jQuery Test Suite

☒ Hide passed tests ☐ Check for Globals ☐ [unclear]

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 Mac OS 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
"`

Trust

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(""));
    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;
```

Maintain

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 than one line" );

    // Check serialization of text values
    equal( jQuery(document.createTextNode("foo")).text(), "foo", "Text node was serialized correctly" );
    notEqual( jQuery(document).text(), "", "Retrieving text for the document returns the correct value" );

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

    equal( jQuery(frag).text(), "foo", "Document Fragment Text node was retrieved correctly" );

    $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 characters" );

    $newLineTest.remove();

});
```

Maintainability Problem

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

    expect( 5 );

    var expected, frag, $newLineTest;

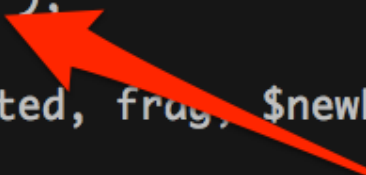
    expected = "This link has class=\"blog\": Simon Willison's Weblog";
    equal( jQuery("#sap").text(), expected, "Check for merged text of more than one line");

    // Check serialization of text values
    equal( jQuery(document.createTextNode("foo")).text(), "foo", "Text node was serialized correctly");
    notEqual( jQuery(document).text(), "", "Retrieving text for the document returns the correct value");

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

    equal( jQuery(frag).text(), "foo", "Document Fragment Text node was retrieved correctly");

    $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 characters");
});
```





John Resig @jeresig 1d

@RoyOshero 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</b>")  
  
    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 = {  
        value: 1,  
        success: true,  
        error: false,  
        futureSuccess: true,  
        futureError: false,  
        notify: true  
    }  
};  
  
    jQuery.when( deferreds ).done( function() {  
        expect( willSucceed ).toEqual( willSucceed );  
    } );  
});
```



```
test( "globalEval with 'use strict'", function() {  
    expect( 1 );  
    Globals.register("strictEvalTest");  
  
    jQuery.globalEval("'use strict'; var strictEval  
    equal( window.strictEvalTest, 1, "Test variable  
});
```


Read

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 more"
```



```
// Check serialization of text values
```

```
equal( jQuery(document.createTextNode("foo")).text(), "foo", "Text no
```


Read

Seperating vars from usage

```
css(String.prototype).css(function() {
    expect( 48 );

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

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

    $child = jQuery("#nothiddendivchild").css({ "width": "200px", "height": "200px" });
    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;'><div style='height:20px;'></div></div>" );

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

    div2.remove();

    // handle negative numbers by setting to zero #11504
    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([0.025, 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("font-size"), 10), 16, "Verify fontSize px set." );
    equal( parseInt(div.css("font-size"), 10), 16, "Verify fontSize px set." );
    equal( parseInt(child.css("font-size"), 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("font-size"), 10), 32, "Verify fontSize em set." );

    // Have to verify this as the result depends upon the browser's CSS
    // support for font-size percentages
    child.attr("class", "text");
    actual = parseInt(child.css("font-size"), 10);
    checkval = 0;

    if ( actual === 16 || actual === 24 ) {
        checkval = actual;
    }

    equal( actual, 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;
```

Read

Comments

```
j = jQuery( "<span>hi</span> there <!-- mon  
ok( j.length >= 2, "Check node,textnode,com  
  
ok( !jQuery("<option>test</option>")[0].se  
  
ok( jQuery("<div></div>")[0], "Create a div  
ok( jQuery("<table></table>")[0], "Create a  
  
// equal( jQuery("element[attribute='<div>  
// equal( jQuery("element[attribute=<div><  
// equal( jQuery("element:not(<div></div>)  
// equal( jQuery("\\<div\\>").length, 0, "  
);
```

Read

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)" );  
});
```

Read

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 then out!";
    result = jQuery("#first").append( valueObj)("debugger/b");

    equal( result.text(), defaultText + "bug", "Check if test 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='radiotest1' type='radio' checked='checked' />");
    jQuery("#form input[name=radiotest1]").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 formatted 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-formatted 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("#dividtest").appendTo("#form").append( valueObj)("div id='legend'>test</div>");
    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'>options</optgroup></options></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'>options</options></select>") );
    t( "Append Select", "AppendSelect1, AppendSelect2", [ "appendSelect1", "appendSelect2" ] );

    equal( "Two nodes", jQuery("<div />").append( "Two", " nodes" ).text(), "Appending two test nodes (#4811)" );
    equal( jQuery("<div />").append( "1", " ", 3 ).text(), "13", "If median is false-like value, subsequent arguments should not be ignored" );

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

    equal( jQuery("#nonodes").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("#nonodes");
    d.remove();
    equal( jQuery("#nonodes").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, "Appending radios uphold which radio is checked" );
    equal( $radioUnchecked[0].checked, false, "Appending radios uphold not being checked" );

    equal( jQuery("<div />").append( valueObj("option<area />")[0].childNodes.length, 2, "HTML-string with leading test 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</b>")  
  
    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" ).append( valueObj( "<input type='radio' checked='checked' name='radio1'" );
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>") ), 1, "Append to iframed document" );
    } 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'" );
```

```
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 an OPTION (all" );
```

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

try {
    if ( iframeDoc && iframeDoc.body ) {
        equal( jQuery(iframeDoc.body).append( valueObj("<div id='success'>test</div>") ), true );
    } 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.append(function( i, val ) {  
        equal( val, old, "Make sure the incoming value is correct." );  
        return "<option value='appendTest'>Append Test</option>";  
    }).find("option:last-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(function( i, val ) {  
        equal( val, old, "Make sure the incoming value is correct." );  
        return document.getElementById("first");  
    });
```

```
    equal( jQuery("#sap").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." );
```



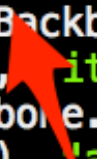
Backbone

Read

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");
});
```



Read

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 variable");
  });
});
```

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', function() {
    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>";
    ko.applyBindings({ myObservable: observable }, testNode);
  });
});

```

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”

Read

Semi-Irrelevant Setup



Batman

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

```
  suite = ->
```



```
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 on
```

```
    @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 start actions does not the complete callba
```

```
    @frame.startOperation()
```

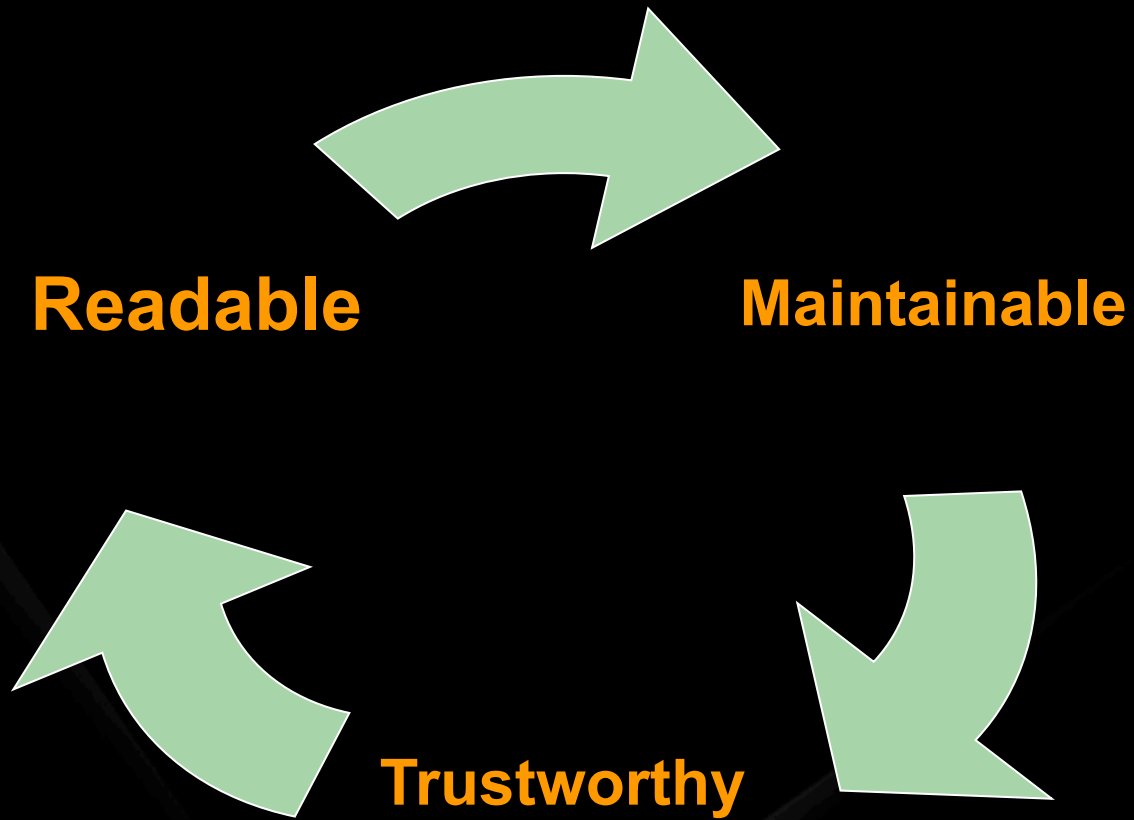
```
    @frame.startOperation()
```

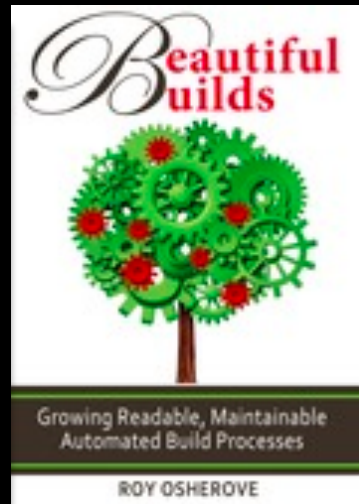


Read

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
```





ArtOfUnitTesting.com
@RoyOsheroove