Thomas Fuchs

Adventures in JavaScript testing

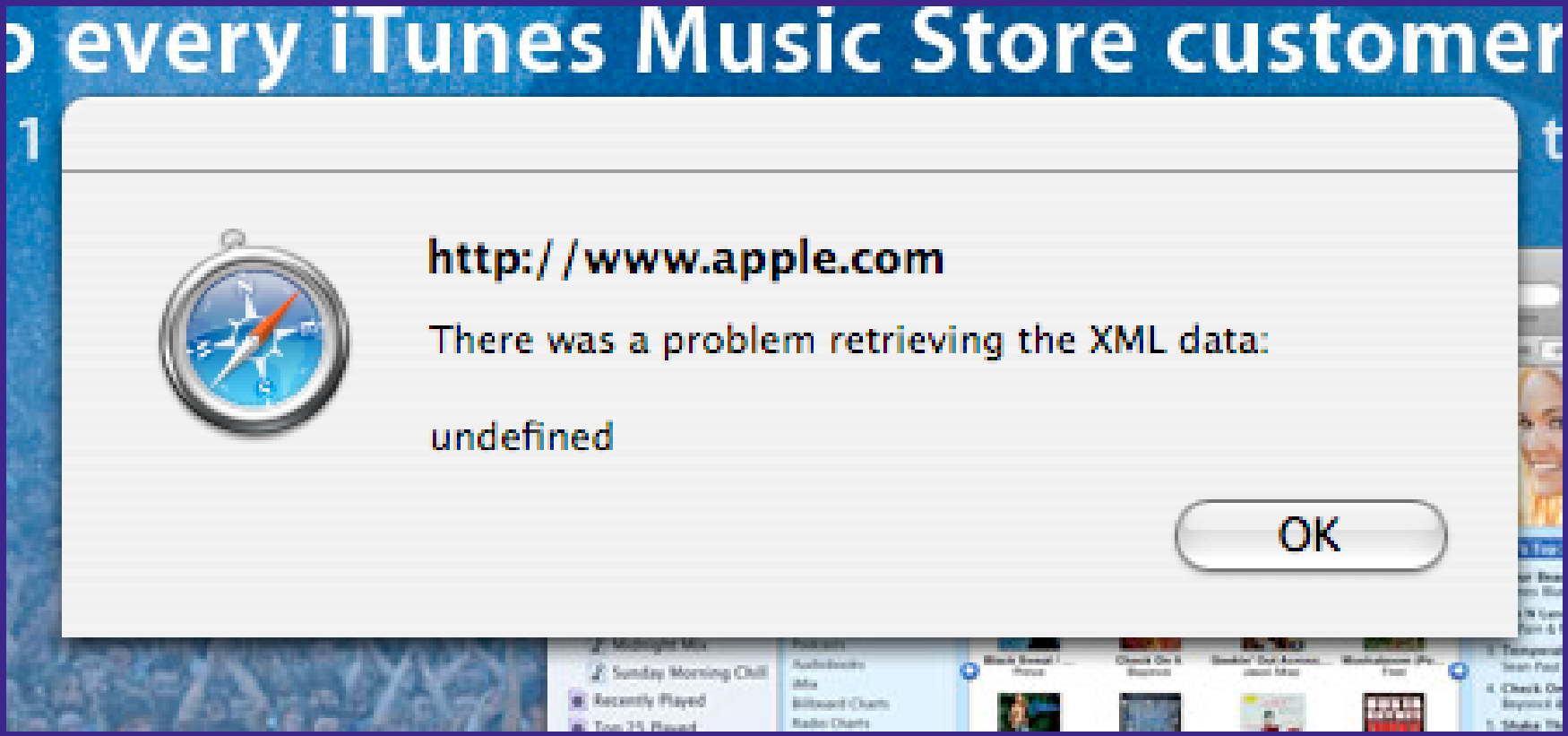
Thomas Fuchs

wollzelle

script.aculo.us

mir.aculo.us

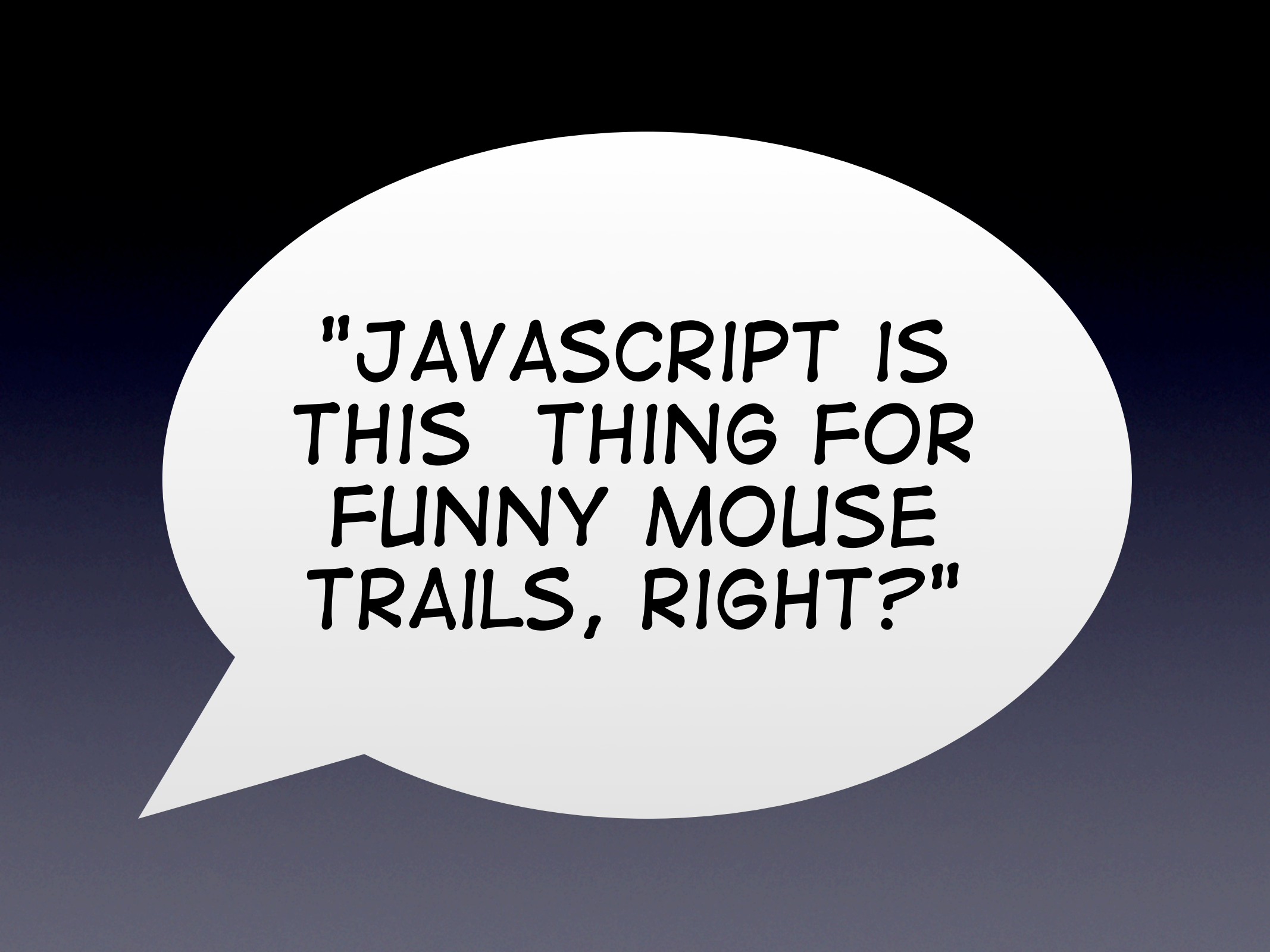
traditional javascript „testing“


every iTunes Music Store customer
http://www.apple.com
There was a problem retrieving the XML data:
undefined
OK

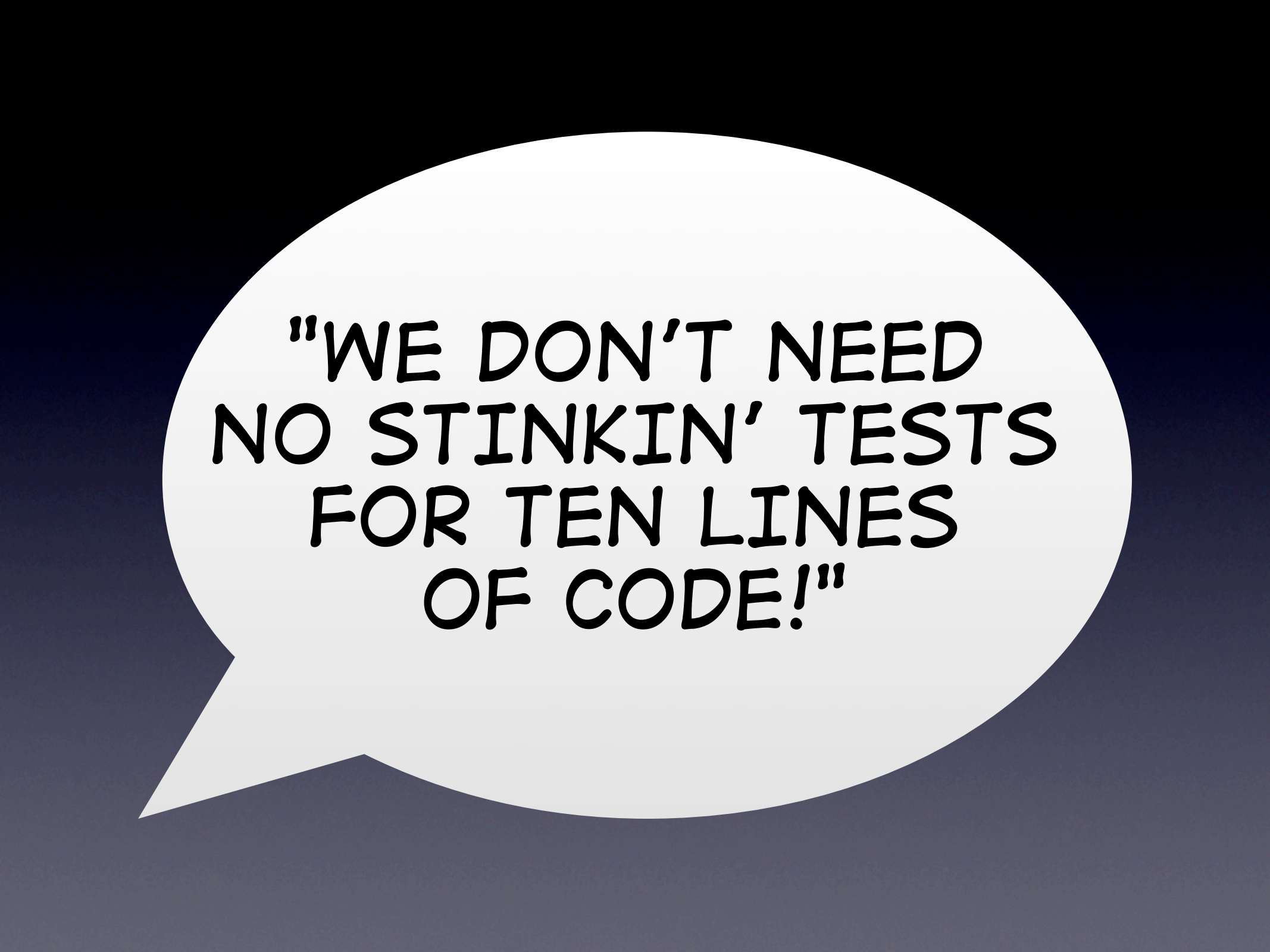
http://www.flickr.com/photos/stephendann/80555607/

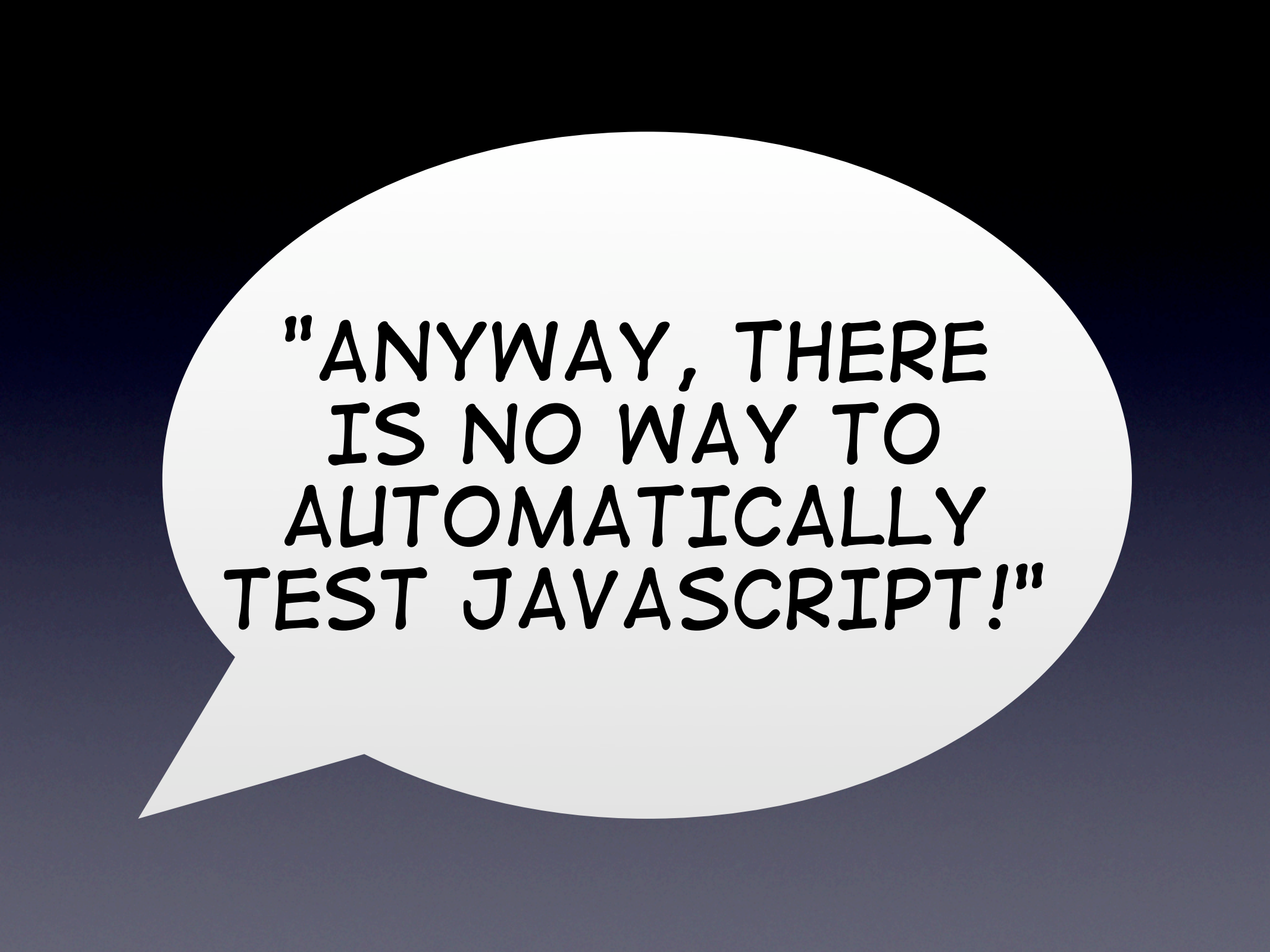
```
alert(“please use me only
  where appropriate!”);
```

why?


"JAVASCRIPT IS THIS THING FOR FUNNY MOUSE TRAILS, RIGHT?"


"BUT YOU
CAN'T DEBUG
JAVASCRIPT!"


"WE DON'T NEED
NO STINKIN' TESTS
FOR TEN LINES
OF CODE!"


"ANYWAY, THERE IS NO WAY TO AUTOMATICALLY TEST JAVASCRIPT!"

all of this is
pure FUD

unit testing javascript

script.aculo.us Builder unit test file

Tests for Builder.

6 tests, 331 assertions, 0 failures, 0 errors

Status	Test	Message
passed	testBuilderBasics	22 assertions, 0 failures, 0 errors
passed	testBuilderAllXHTMLTags	284 assertions, 0 failures, 0 errors
passed	testBuilderOptionTag	6 assertions, 0 failures, 0 errors
passed	testBuilderContatenation	15 assertions, 0 failures, 0 errors
passed	testBuilderComplexExample	2 assertions, 0 failures, 0 errors
passed	testBuilderShortcuts	2 assertions, 0 failures, 0 errors

```
<html>
<head>
  <title>script.aculo.us Unit test file</title>
  <script src="../../lib/prototype.js" type="text/javascript"></script>
  <script src="../../src/scriptaculous.js" type="text/javascript"></script>
  <script src="../../src/unittest.js" type="text/javascript"></script>
  <link rel="stylesheet" href="../test.css" type="text/css" />
</head>
<body>
<!-- Log output -->
<div id="testlog"> </div>

<!-- Tests follow -->
<script type="text/javascript" language="javascript" charset="utf-8">
// <![CDATA[

  new Test.Unit.Runner({

    setup: function() {
      // setup runs before each test
    },

    teardown: function() {
      // teardown runs after each test
    },

    testBuilderBasics: function() { with(this) {
      var element = Builder.node('div');
      assertEqual('DIV', element.nodeName);
    }}

  });
// ]]>
</script>
</body>
</html>
```

Borrows from Test::Unit

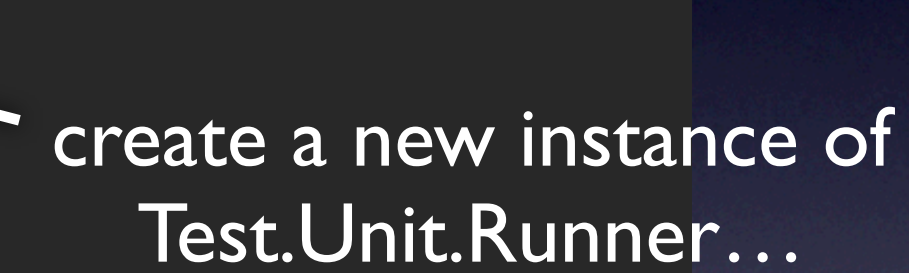
```
<html>
<head>
  <title>script.aculo.us Unit test file</title>
  <script src="../../lib/prototype.js" type="text/javascript"></script>
  <script src="../../src/scriptaculous.js" type="text/javascript"></script>
  <script src="../../src/unittest.js" type="text/javascript"></script>
  <link rel="stylesheet" href="../test.css" type="text/css" />
</head>
<body>
<!-- Log output -->
<div id="testlog"> </div>

<!-- Tests follow -->
<script type="text/javascript" language="javascript" charset="utf-8">
// <![CDATA[

  new Test.Unit.Runner({

    setup: function() {
      // setup runs before each test
    },

    teardown: function() {
      // teardown runs after each test
    },

    testBuilderBasics: function() { with(this) {
      var element = Builder.node('div');
      assertEqual('DIV', element.nodeName);
    }}

  });
// ]]>
</script>
</body>
</html>
```

include unittest.js
(requires Prototype)

```
<html>
<head>
  <title>script.aculo.us Unit test file</title>
  <script src="../../lib/prototype.js" type="text/javascript"></script>
  <script src="../../src/scriptaculous.js" type="text/javascript"></script>
  <script src="../../src/unittest.js" type="text/javascript"></script>
  <link rel="stylesheet" href="../test.css" type="text/css" />
</head>
<body>
<!-- Log output -->
<div id="testlog"> </div>

<!-- Tests follow -->
<script type="text/javascript" language="javascript" charset="utf-8">
// <![CDATA[

  new Test.Unit.Runner({

    setup: function() {
      // setup runs before each test
    },

    teardown: function() {
      // teardown runs after each test
    },

    testBuilderBasics: function() { with(this) {
      var element = Builder.node('div');
      assertEqual('DIV', element.nodeName);
    }}

  });
// ]]>
</script>
</body>
</html>
```

a DIV is required for containing the test results (defaults to "testlog")

```
<html>
<head>
  <title>script.aculo.us Unit test file</title>
  <script src="../../lib/prototype.js" type="text/javascript"></script>
  <script src="../../src/scriptaculous.js" type="text/javascript"></script>
  <script src="../../src/unittest.js" type="text/javascript"></script>
  <link rel="stylesheet" href="../test.css" type="text/css" />
</head>
<body>
<!-- Log output -->
<div id="testlog"> </div>

<!-- Tests follow -->
<script type="text/javascript" language="javascript" charset="utf-8">
// <![CDATA[

  new Test.Unit.Runner({

    setup: function() {
      // setup runs before each test
    },

    teardown: function() {
      // teardown runs after each test
    },

    testBuilderBasics: function() { with(this) {
      var element = Builder.node('div');
      assertEqual('DIV', element.nodeName);
    }}

  });
// ]]>
</script>
</body>
</html>
```



```
<html>
<head>
  <title>script.aculo.us Unit test file</title>
  <script src="../../lib/prototype.js" type="text/javascript"></script>
  <script src="../../src/scriptaculous.js" type="text/javascript"></script>
  <script src="../../src/unittest.js" type="text/javascript"></script>
  <link rel="stylesheet" href="../test.css" type="text/css" />
</head>
<body>
<!-- Log output -->
<div id="testlog"> </div>

<!-- Tests follow -->
<script type="text/javascript" language="javascript" charset="utf-8">
// <![CDATA[

  new Test.Unit.Runner({

    setup: function() {
      // setup runs before each test
    },

    teardown: function() {
      // teardown runs after each test
    },

    testBuilderBasics: function() { with(this) {
      var element = Builder.node('div');
      assertEqual('DIV', element.nodeName);
    }}

  });
// ]]>
</script>
</body>
</html>
```

…this will excute all methods starting with "test"

```
<html>
<head>
  <title>script.aculo.us Unit test file</title>
  <script src="../../lib/prototype.js" type="text/javascript"></script>
  <script src="../../src/scriptaculous.js" type="text/javascript"></script>
  <script src="../../src/unittest.js" type="text/javascript"></script>
  <link rel="stylesheet" href="../test.css" type="text/css" />
</head>
<body>
<!-- Log output -->
<div id="testlog"> </div>

<!-- Tests follow -->
<script type="text/javascript" language="javascript" charset="utf-8">
// <![CDATA[

  new Test.Unit.Runner({

    setup: function() {
      // setup runs before each test
    },

    teardown: function() {
      // teardown runs after each test
    },

    testBuilderBasics: function() { with(this) {
      var element = Builder.node('div');
      assertEqual('DIV', element.nodeName);
    }}

  });
// ]]>
</script>
</body>
</html>
```

```
<html>
<head>
  <title>script.aculo.us Unit test file</title>
  <script src="../../lib/prototype.js" type="text/javascript"></script>
  <script src="../../src/scriptaculous.js" type="text/javascript"></script>
  <script src="../../src/unittest.js" type="text/javascript"></script>
  <link rel="stylesheet" href="../test.css" type="text/css" />
</head>
<body>
<!-- Log output -->
<div id="testlog"> </div>

<!-- Tests follow -->
<script type="text/javascript" language="javascript" charset="utf-8">
// <![CDATA[

  new Test.Unit.Runner({

    setup: function() {
      // setup runs before each test
    },

    teardown: function() {
      // teardown runs after each test
    },

    testBuilderBasics: function() { with(this) {
      var element = Builder.node('div');
      assertEqual('DIV', element.nodeName);
    }}

  });
// ]]>
</script>
</body>
</html>
```

(syntax strangeness for more convenience when calling assertions)

assert(expression)

```
assert(true) => PASS
assert(false) => FAIL
assert(2*2==4) => PASS
```

```
assertEqual(expected, actual)
```

```
assertEqual('a','a') => PASS
assertEqual('a','b') => FAIL
  assertEqual(1,1) => PASS
 assertEqual(1,'1') => PASS
```

does not compare type

```
assertEnumEqual(expected, actual)
```

```
assertEnumEqual([1,2],[1,2]) => PASS
```

```
assertEqual([1,2],[1,2]) => FAIL
```

assertEqual does not
compare enum contents

```
assertNotEqual(expected, actual)
```

```
assertNotEqual('a','a') => FAIL
assertNotEqual('a','b') => PASS
  assertNotEqual(1,1) => FAIL
 assertNotEqual(1,'1') => FAIL
```

does not compare type

assertMatch(expected, actual)

```
assertMatch(/a/,'a') => PASS
assertMatch(/a/,'bab') => PASS
assertMatch(/a/,'b') => FAIL
assertMatch(/^moo$/,'moo') => PASS
```

```
assertIdentical(expected, actual)
```

```
assertIdentical('a','a') => PASS
assertIdentical(1,'1') => FAIL
```

compares type

```
assertNotIdentical(expected, actual)
```

```
assertNotIdentical('a','a') => FAIL
 assertNotIdentical(1,'1') => PASS
```

compares type

```
assertType(expected, actual)
```

```
assertType(String,'a') => PASS
assertType(Number,1) => PASS
assertType(Array,[1,2]) => PASS
```

checks for a specific constructor

assertRaise(exceptionName, method)

```
assertRaise('ElementDoesNotExistError',
   function(){
     new Effect.Opacity('invalid-element')
   }
);
```

related effects.js code

```
if(!this.element) throw(Effect._elementDoesNotExistError);
```

```
var Effect = {
  _elementDoesNotExistError: {
    name: 'ElementDoesNotExistError',
    message: 'The specified DOM element does not exist, but is required for this effect to operate'
  },
```

assertRespondsTo(method, obj)

```
var someObject = {
  doSomething: function(){
    // ...
  }
}
```

```
assertRespondsTo('doSomething', someObject);
```

```
assertVisible(element)
```

```
assertNotVisible(element)
```

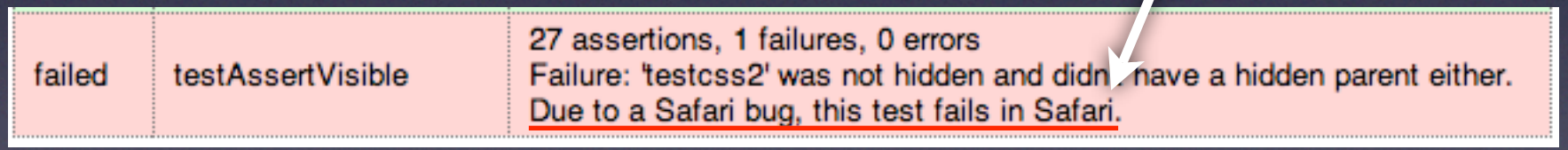
also checks if any parent elements are hidden and fails/passes on that

```
info(message)
```

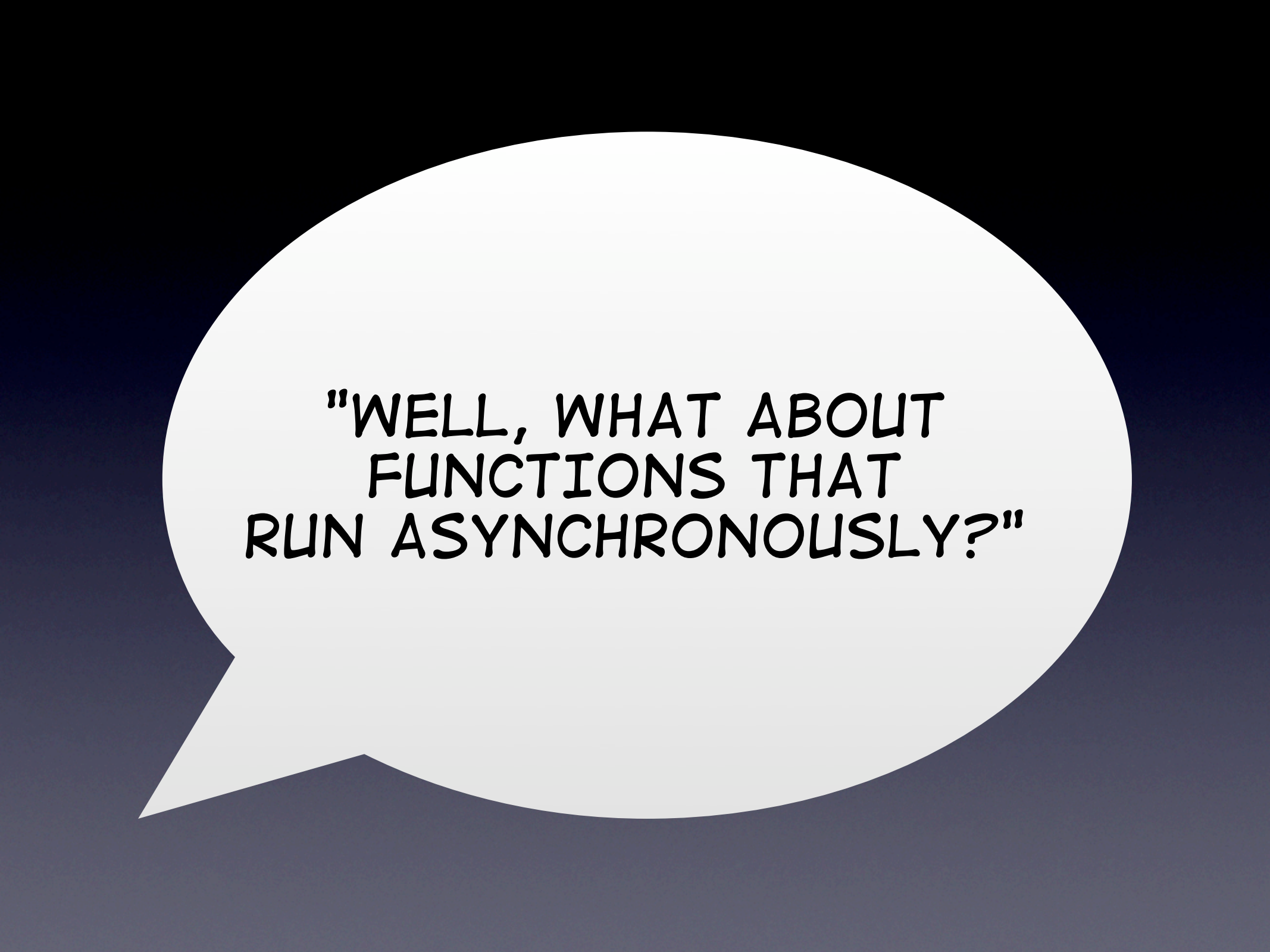
Displays arbitrary
messages in the "Message"
column of the test result

assertXYZ(params, message)

```
assertNotVisible('testcss2',
  "Due to a Safari bug, this test fails in Safari.");
```



failed	testAssertVisible	27 assertions, 1 failures, 0 errors Failure: 'testcss2' was not hidden and didn't have a hidden parent either. Due to a Safari bug, this test fails in Safari.


"WELL, WHAT ABOUT FUNCTIONS THAT RUN ASYNCHRONOUSLY?"

wait(milliseconds, method)

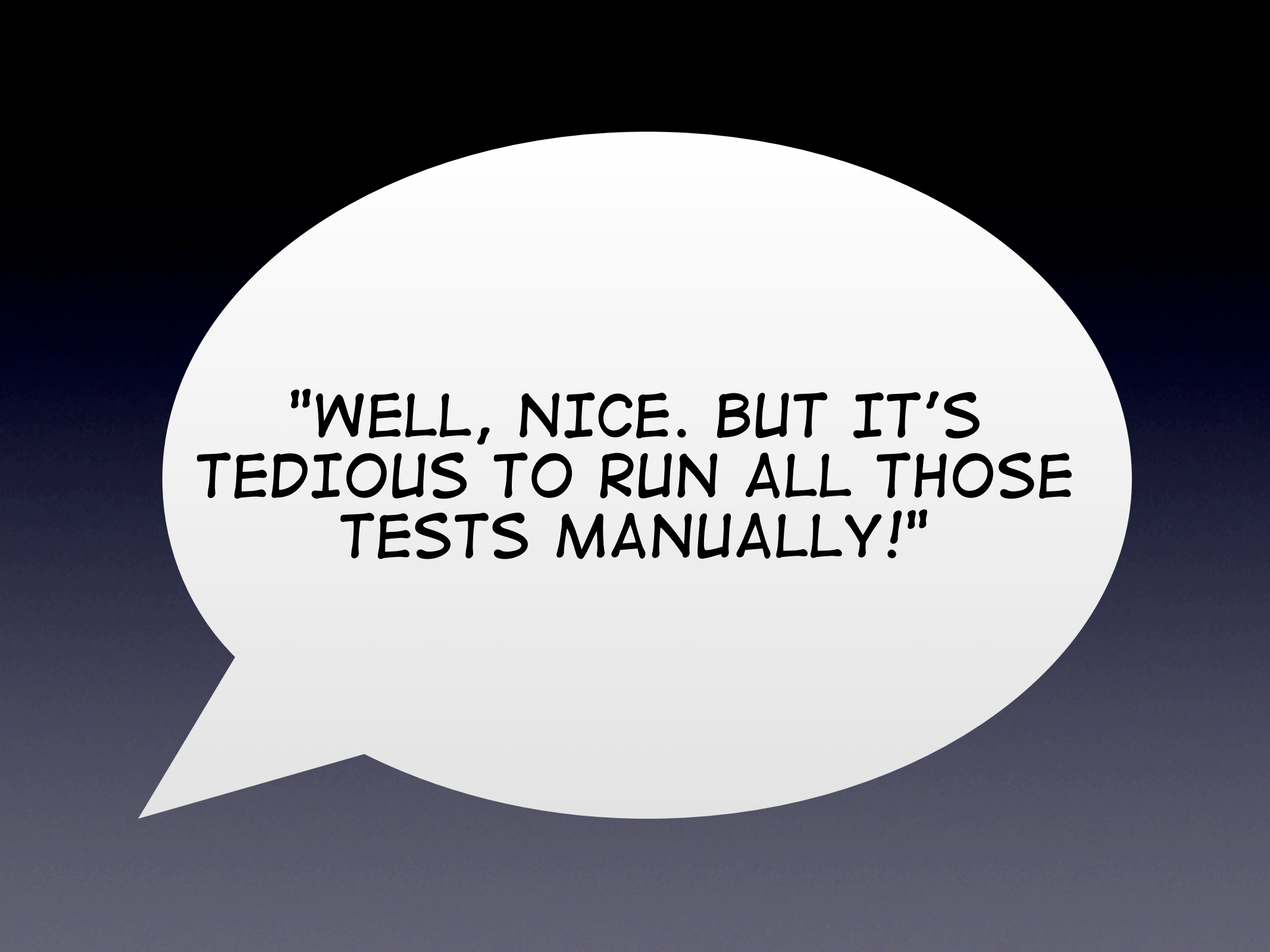
```
testInspect: function() { with(this) {
  var e1 = new Effect.Opacity('sandbox',{from:1.0,to:0.5,duration:0.5});
  assertEqual(0, e1.inspect().indexOf('#<Effect:'));
  assert(e1.inspect().indexOf('idle')>0);
  wait(1000, function() {
    assert(e1.inspect().indexOf('finished')>0);
  });
}},
```

should be last statement in the test (but can be nested)

Benchmarking

```
benchmark(function(){
  Element.childrenWithClassName("Container", "firstClass")
},1000,'Element.childrenWithClassName');
```

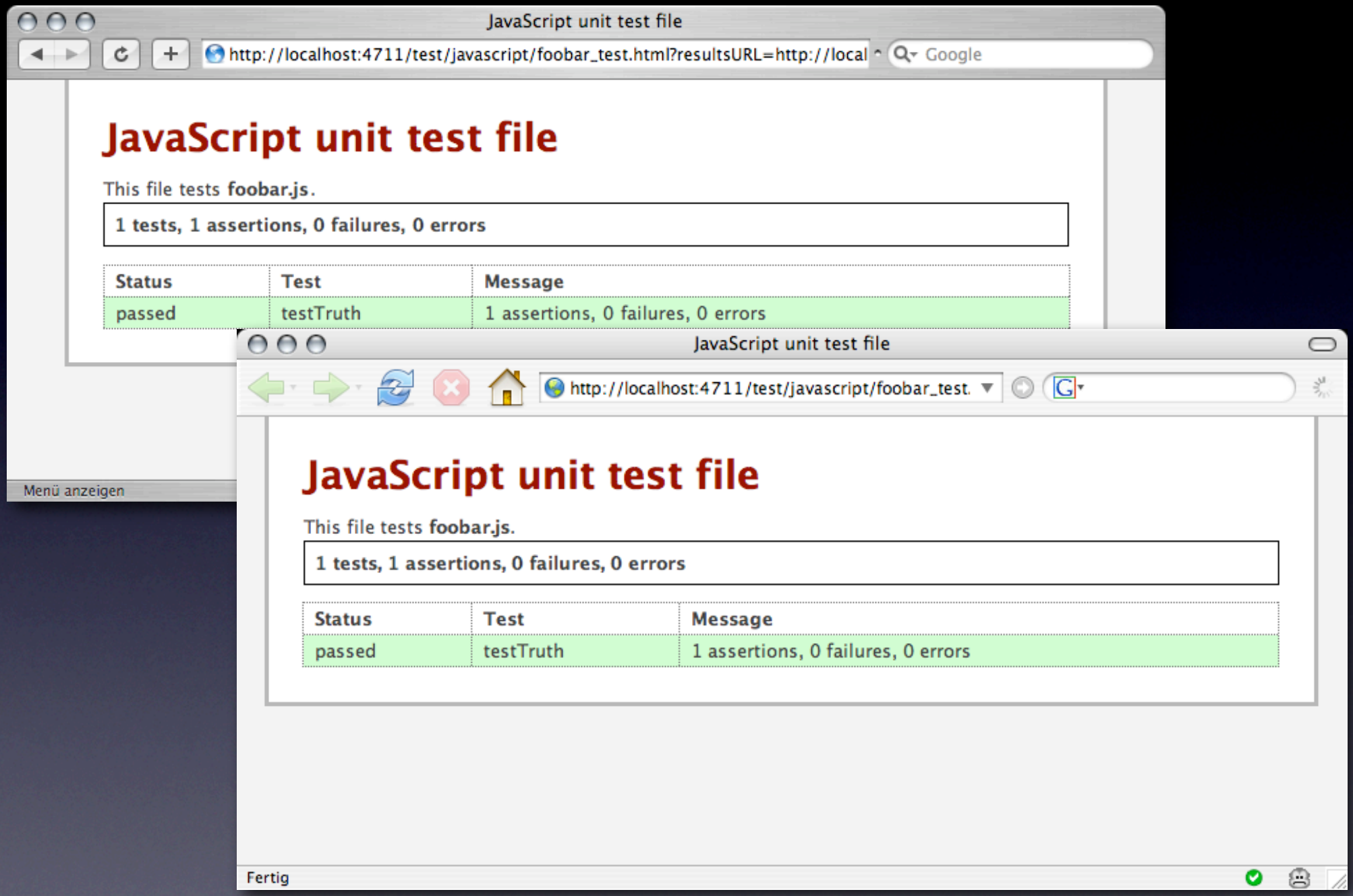
passed	testElementChildrenWithClassName	10 assertions, 0 failures, 0 errors Info: Element.childrenWithClassName finished 1000 iterations in 1.254s


"WELL, NICE. BUT IT'S TEDIOUS TO RUN ALL THOSE TESTS MANUALLY!"

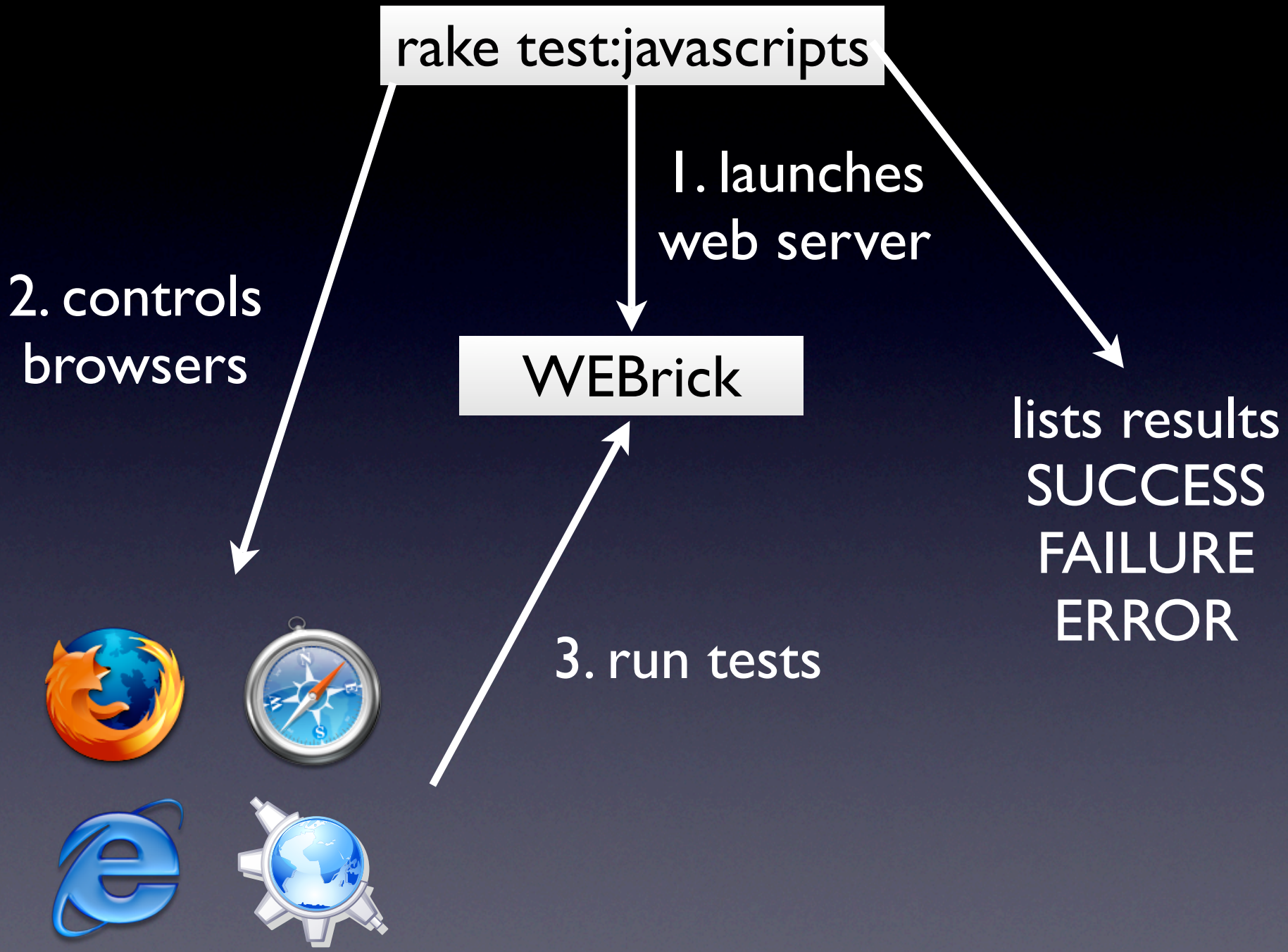
rake to the rescue

```
rake test:javascripts
```

```
$ rake test:javascripts
(in /Users/thomasfuchs/Web/fluxiom-dev)
/test/javascript/fluxiom_test.html on Safari: SUCCESS
/test/javascript/fluxiom_test.html on Firefox: SUCCESS
Skipping Internet Explorer, not supported on this OS
Skipping Konqueror, not supported on this OS
```


JavaScript unit test file
http://localhost:4711/test/javascript/foobar_test.html?resultsURL=http://local
Google
JavaScript unit test file
This file tests foobar.js.
1 tests, 1 assertions, 0 failures, 0 errors
Status
Test
Message
passed
testTruth
1 assertions, 0 failures, 0 errors
Menü anzeigen
JavaScript unit test file
http://localhost:4711/test/javascript/foobar_test.
JavaScript unit test file
This file tests foobar.js.
1 tests, 1 assertions, 0 failures, 0 errors
Status
Test
Message
passed
testTruth
1 assertions, 0 failures, 0 errors
Fertig

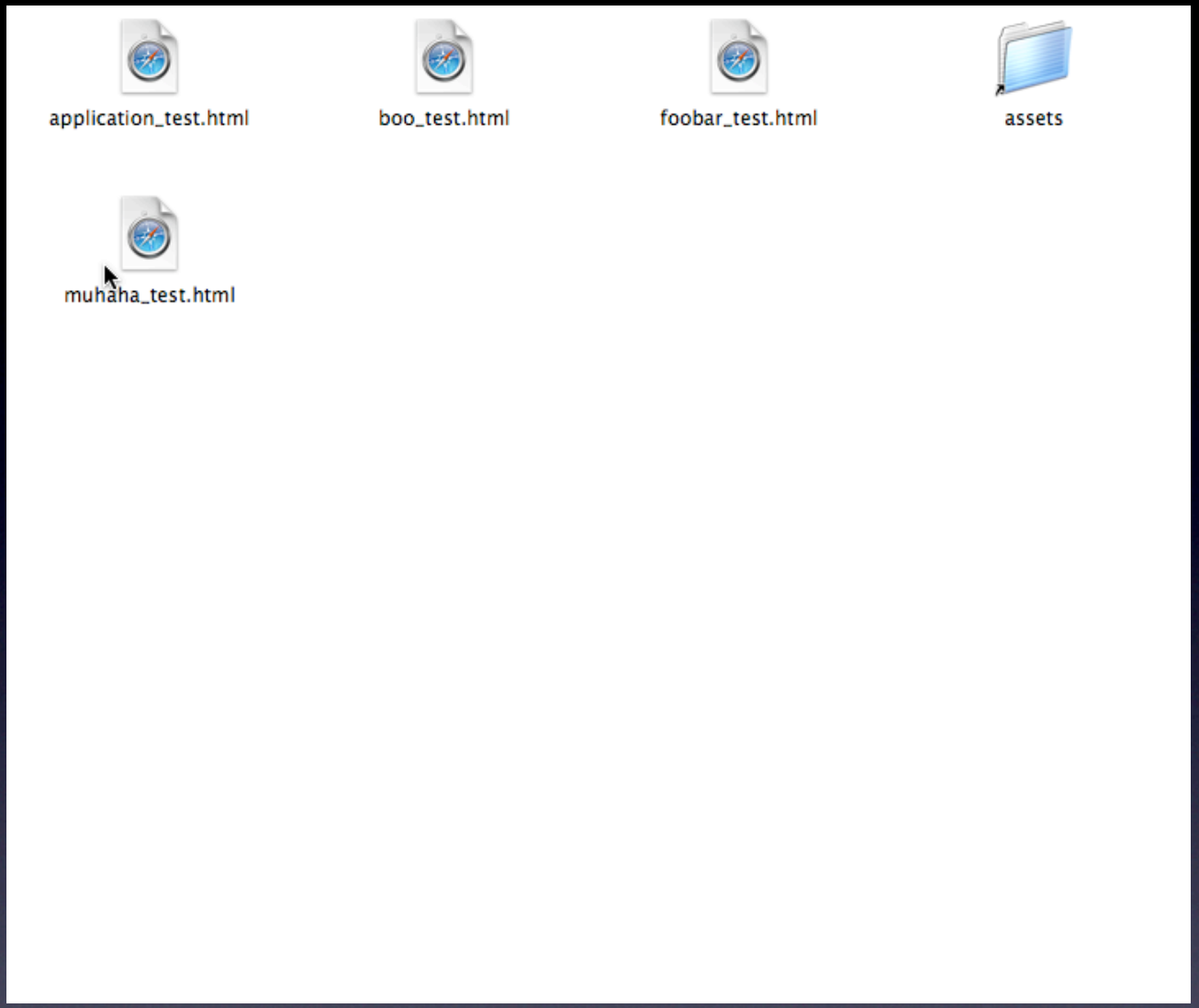
javascript_test plugin


rake test:javascripts
1. launches web server
2. controls browsers
WEBrick
lists results
SUCCESS
FAILURE
ERROR
3. run tests

```
$ script/generate javascript_test muhaha
  exists  test/javascript
  create  test/javascript/muhaha_test.html
```

just
add your
tests

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>JavaScript unit test file</title>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <script src="assets/prototype.js" type="text/javascript"></script>
  <script src="assets/unittest.js" type="text/javascript"></script>

  <script src="../../public/javascripts/muhaha.js" type="text/javascript"></script>

  <link rel="stylesheet" href="assets/unittest.css" type="text/css" />
</head>
<body>

<div id="content">

  <div id="header">
    <h1>JavaScript unit test file</h1>
    <p>
      This file tests <strong>muhaha.js</strong>.
    </p>
  </div>

  <!-- Log output -->
  <div id="testlog"> </div>

</div>

<script type="text/javascript">
// <![CDATA[

  new Test.Unit.Runner({

    // replace this with your real tests

    setup: function() {

    },

    teardown: function() {

    },

    testTruth: function() { with(this) {
      assert(true);
    }}

  }, "testlog");
// ]]>
</script>
</body>
</html>
```



(symlink required:
app/test/javascript/assets to
app/vendor/plugins/javascript_test/assets)

Browser definitions

(add more!)

```
class FirefoxBrowser < Browser¬
  def initialize(path='c:\Program Files\Mozilla Firefox\firefox.exe')¬
    @path = path¬
  end¬
¬
  def visit(url)¬
    applescript('tell application "Firefox" to Get URL "' + url + '"') if macos? ¬
    system("#{@path} #{url}") if windows? ¬
    system("firefox #{url}") if linux?¬
  end¬
¬
  def to_s¬
    "Firefox"¬
  end¬
end¬
```

```
$ script/plugin install http://dev.rubyonrails.org/svn/rails/plugins/javascript_test
```

…and what about BDD?

http://www.flickr.com/photos/sfllaw/4281980/

```
Test.context("BDD-style testing",{
  'should provide extensions to tie in isSomething object methods': function(){
    Object.extend(testObj, Test.BDDMethods);

    testObj.shouldBe('nice');
    testObj.shouldNotBe('broken');
  },

  'should automatically add extensions to strings': function(){
    'a'.shouldEqual('a');
    'a'.shouldNotEqual('b');
    'a'.shouldNotBeNull();
    'a'.shouldBeA(String);

    var aString = 'boo!';
    aString.shouldEqual('boo!');
    aString.shouldBeA(String);
    aString.shouldNotBeA(Number);
  }
});
```

Borrows from RSpec

specifications

```
Test.context("BDD-style testing",{
  'should provide extensions to tie in isSomething object methods': function(){
    Object.extend(testObj, Test.BDDMethods);

    testObj.shouldBe('nice');
    testObj.shouldNotBe('broken');
  },

  'should automatically add extensions to strings': function(){
    'a'.shouldEqual('a');
    'a'.shouldNotEqual('b');
    'a'.shouldNotBeNull();
    'a'.shouldBeA(String);

    var aString = 'boo!';
    aString.shouldEqual('boo!');
    aString.shouldBeA(String);
    aString.shouldNotBeA(Number);
  }
});
```

```
Test.context("BDD-style testing",{
  'should provide extensions to tie in isSomething object methods': function(){
    Object.extend(testObj, Test.BDDMethods);

    testObj.shouldBe('nice');
    testObj.shouldNotBe('broken');
  },

  'should automatically add extensions to strings': function(){
    'a'.shouldEqual('a');
    'a'.shouldNotEqual('b');
    'a'.shouldNotBeNull();
    'a'.shouldBeA(String);

    var aString = 'boo!';
    aString.shouldEqual('boo!');
    aString.shouldBeA(String);
    aString.shouldNotBeA(Number);
  }
});
```

add BDD methods to any object

```
Test.context("BDD-style testing",{
  'should provide extensions to tie in isSomething object methods': function(){
    Object.extend(testObj, Test.BDDMethods);

    testObj.shouldBe('nice');
    testObj.shouldNotBe('broken');
  },

  'should automatically add extensions to strings': function(){
    'a'.shouldEqual('a');
    'a'.shouldNotEqual('b');
    'a'.shouldNotBeNull();
    'a'.shouldBeA(String);

    var aString = 'boo!';
    aString.shouldEqual('boo!');
    aString.shouldBeA(String);
    aString.shouldNotBeA(Number);
  }
});
```

for isSomething() methods, that return a boolean

```
Test.context("BDD-style testing",{
  'should provide extensions to tie in isSomething object methods': function(){
    Object.extend(testObj, Test.BDDMethods);

    testObj.shouldBe('nice');
    testObj.shouldNotBe('broken');
  },

  'should automatically add extensions to strings': function(){
    'a'.shouldEqual('a');
    'a'.shouldNotEqual('b');
    'a'.shouldNotBeNull();
    'a'.shouldBeA(String);

    var aString = 'boo!';
    aString.shouldEqual('boo!');
    aString.shouldBeA(String);
    aString.shouldNotBeA(Number);
  }
});
```

should* assertations,
directly on objects

```
Test.context("BDD-style testing",{
  'should provide extensions to tie in isSomething object methods': function(){
    Object.extend(testObj, Test.BDDMethods);

    testObj.shouldBe('nice');
    testObj.shouldNotBe('broken');
  },

  'should automatically add extensions to strings': function(){
    'a'.shouldEqual('a');
    'a'.shouldNotEqual('b');
    'a'.shouldNotBeNull();
    'a'.shouldBeA(String);

    var aString = 'boo!';
    aString.shouldEqual('boo!');
    aString.shouldBeA(String);
    aString.shouldNotBeA(Number);
  }
});
```

("with" ugliness no longer required)

script.aculo.us Unit test file

BDD-style testing: 10 tests, 47 assertions, 0 failures, 0 errors

Status	Test	Message
passed	-> should run setup before each specification	2 assertions, 0 failures, 0 errors
passed	-> should run teardown after each specification	1 assertions, 0 failures, 0 errors
passed	-> should provide extensions to tie in isSomething object methods	2 assertions, 0 failures, 0 errors
passed	-> should automatically add extensions to strings	7 assertions, 0 failures, 0 errors
passed	-> should automatically add extensions to numbers	4 assertions, 0 failures, 0 errors
passed	-> should automatically add extensions to arrays	3 assertions, 0 failures, 0 errors
passed	-> should support the eval() method	1 assertions, 0 failures, 0 errors
passed	-> should support equality assertion	5 assertions, 0 failures, 0 errors
passed	-> should provide all assertions	19 assertions, 0 failures, 0 errors
passed	-> should support deferred execution	3 assertions, 0 failures, 0 errors

Mapping

```
var METHODMAP = {
  shouldEqual:        'assertEqual',
  shouldNotEqual:     'assertNotEqual',
  shouldEqualEnum:    'assertEnumEqual',
  shouldBeA:          'assertType',
  shouldNotBeA:       'assertNotOfType',
  shouldBeAn:         'assertType',
  shouldNotBeAn:      'assertNotOfType',
  shouldBeNull:       'assertNull',
  shouldNotBeNull:    'assertNotNull',
  shouldBe:           'assertReturnsTrue',
  shouldNotBe:        'assertReturnsFalse'
};
```

Mixed environment

```
'should automatically add extensions to arrays': function(){
  ['a'].shouldNotBeA(String);
  [1,2,3].shouldBeAn(Array);
  [1,2,3].shouldEqualEnum([1,2,3]);
},

'should support the eval() method': function(){
  eval('2*2').shouldEqual(4);
},

'should support equality assertion': function(){
  assertEqual(1, 1);
  assertEqual('a', 'a');
  assertEqual(1, '1');

  var x = 1;
  var y = 1;
  assertEqual(1, x)
  assertEqual(x, y);
},
```

can also use
"normal" assertations

…and if something breaks?

≈

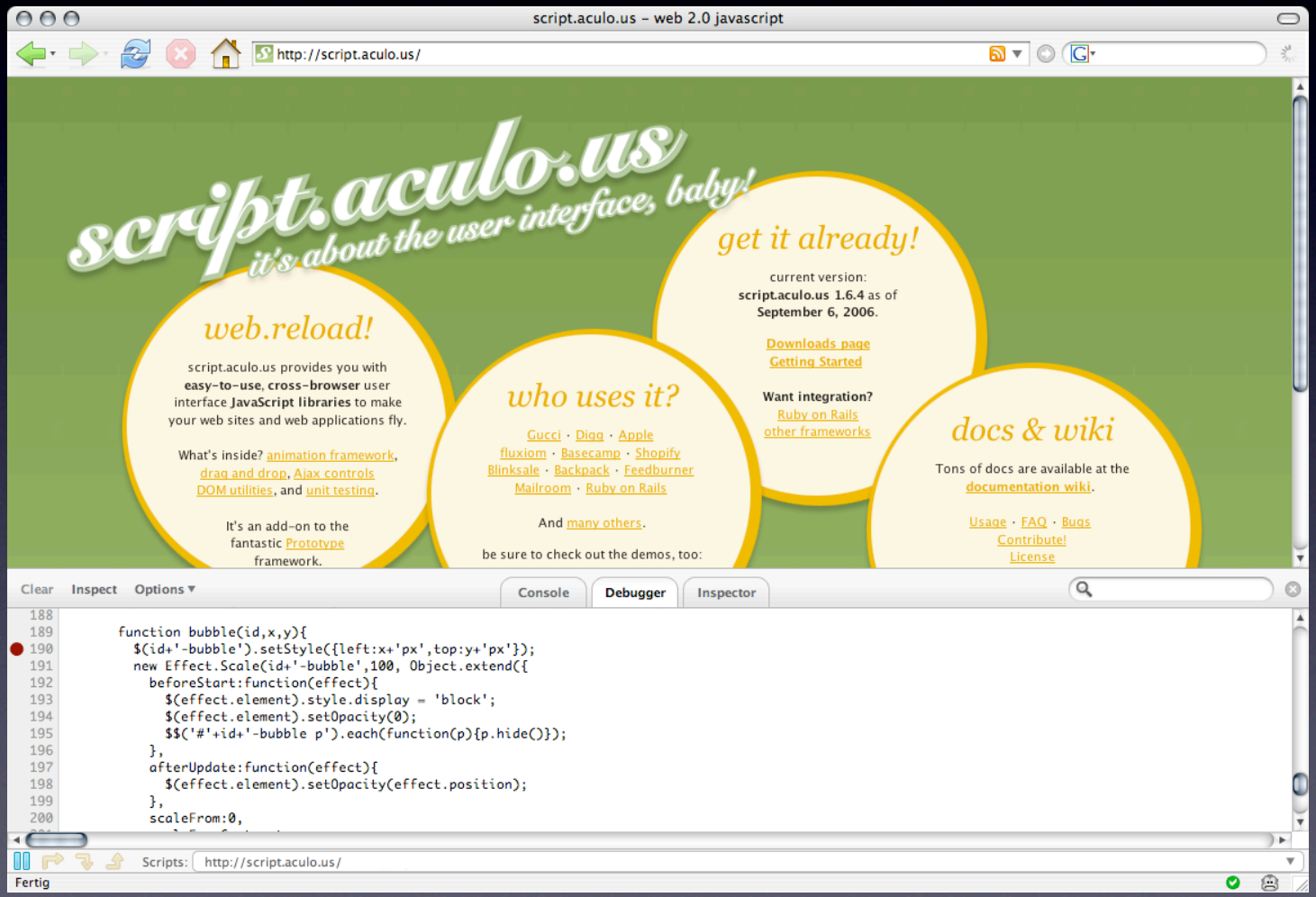
Firebug

sliced bread

http://www.flickr.com/photos/teagrrl/78941282/

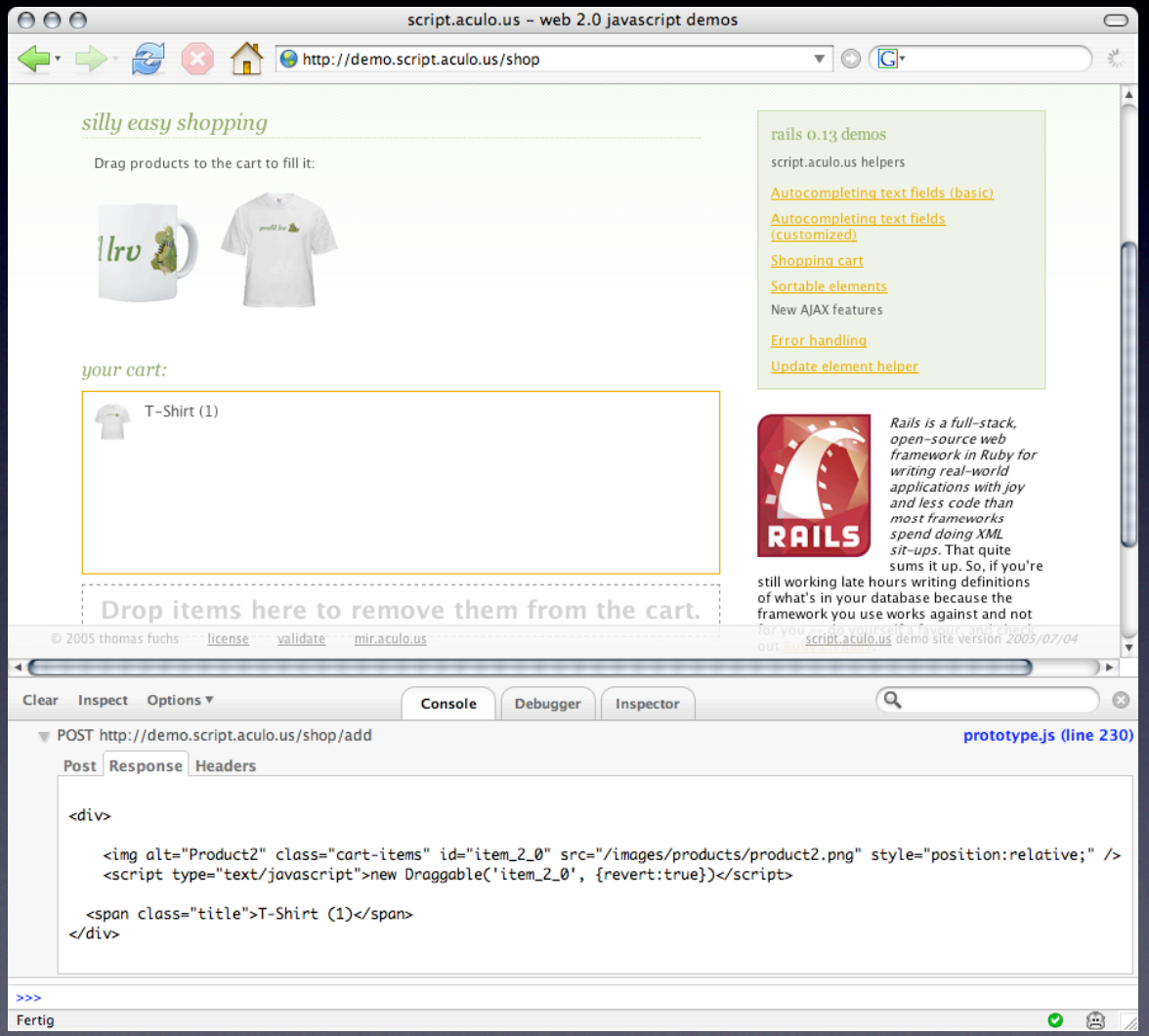
Firebug

Debugger

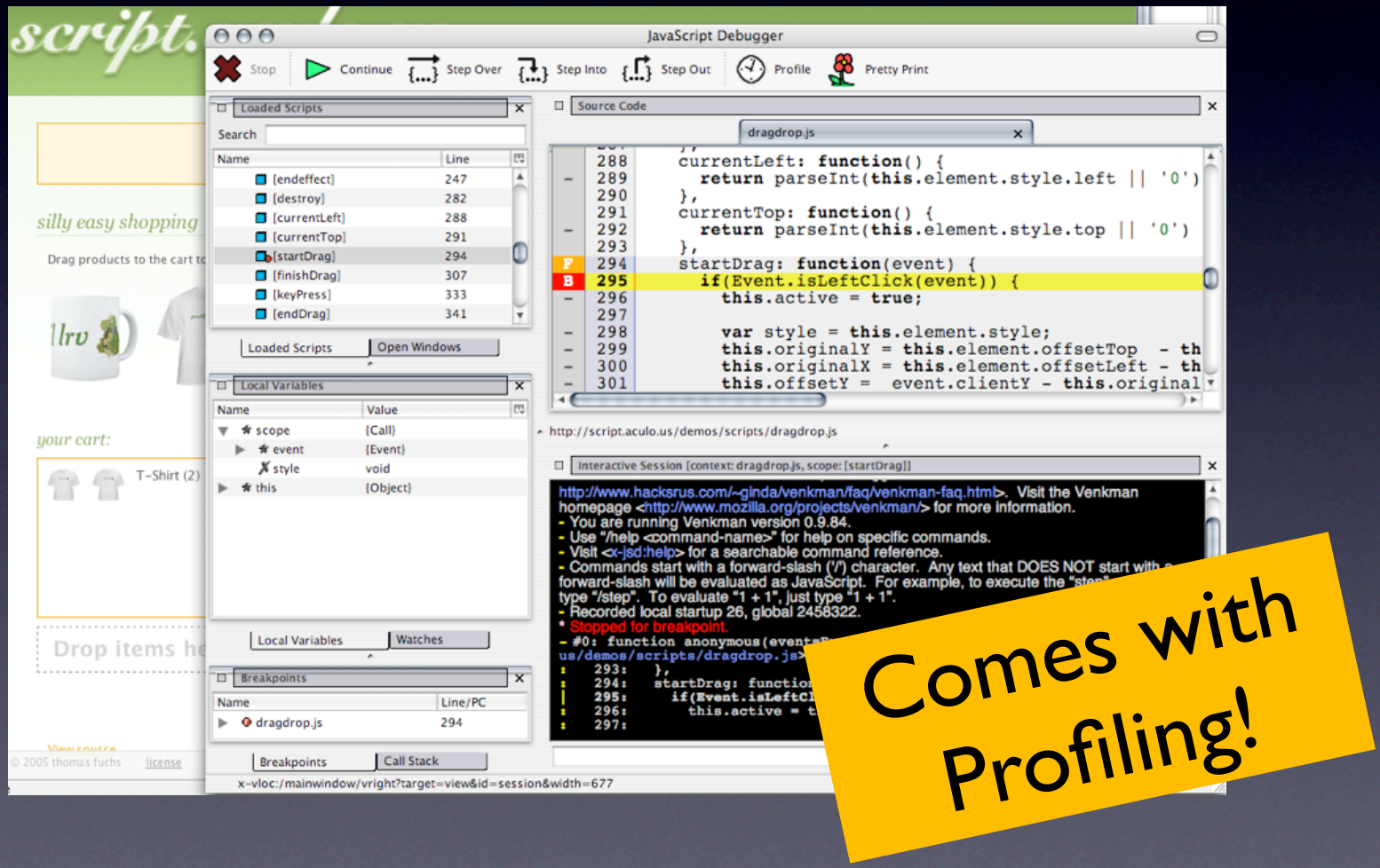


Firebug

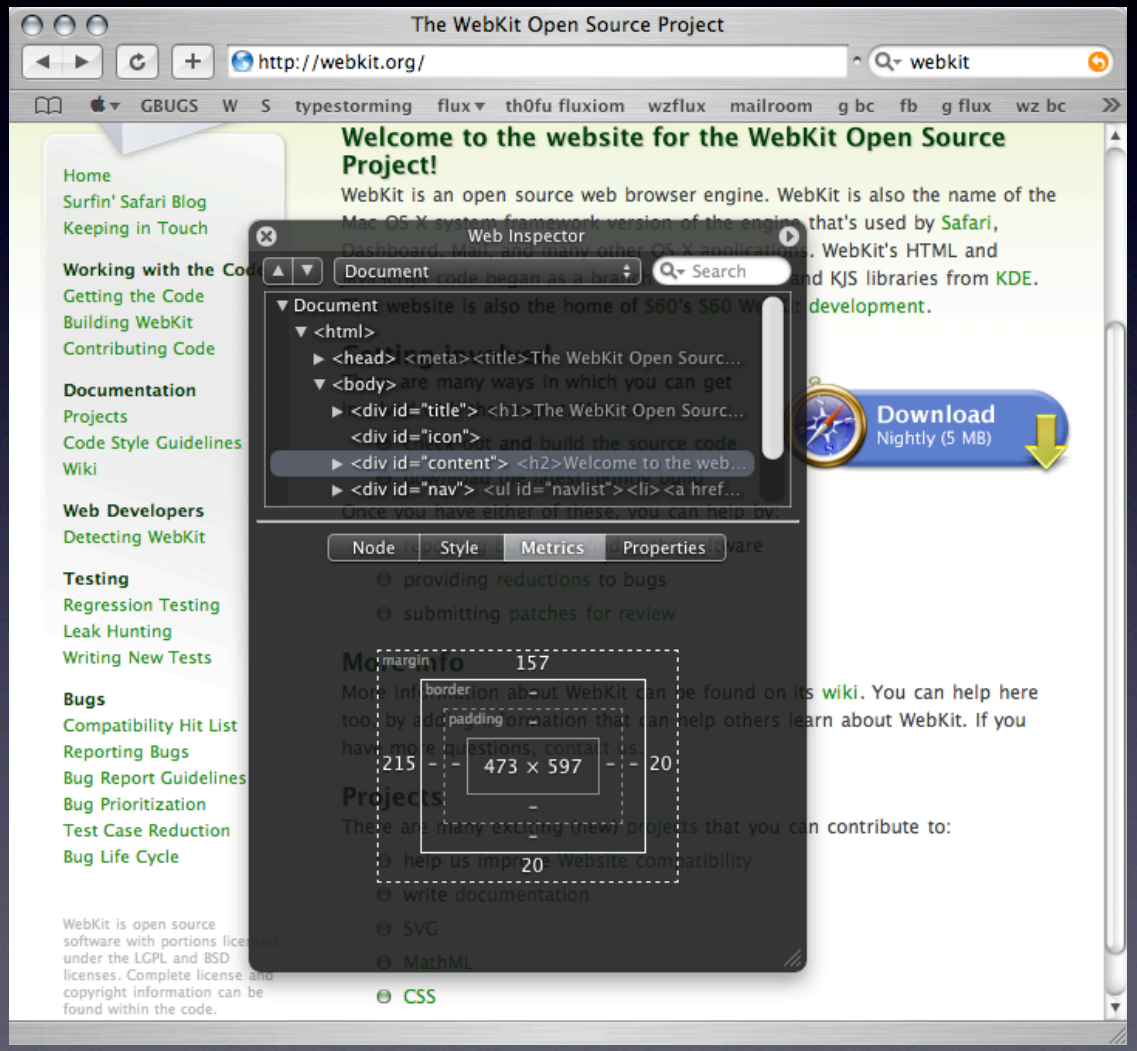
know your Ajax

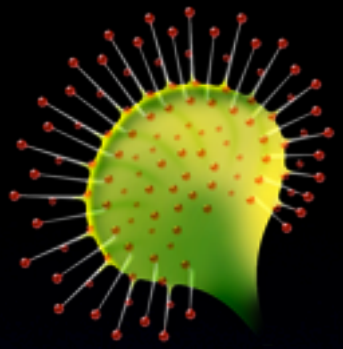


Venkman JavaScript Debugger

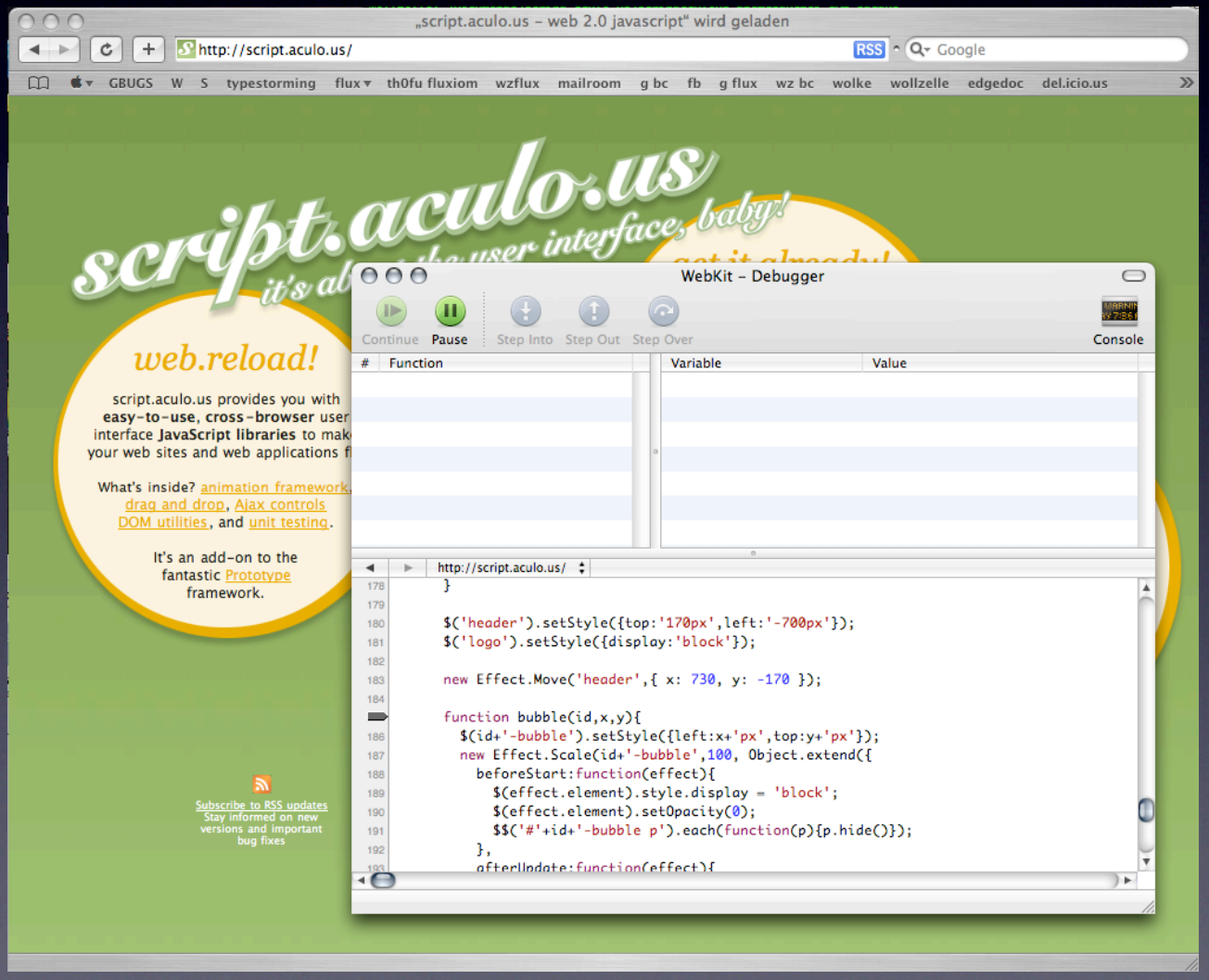


Safari Web Inspector

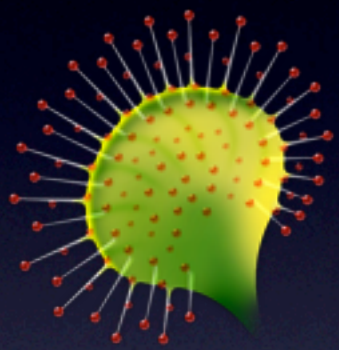




Safari Debugging with Drosera



+



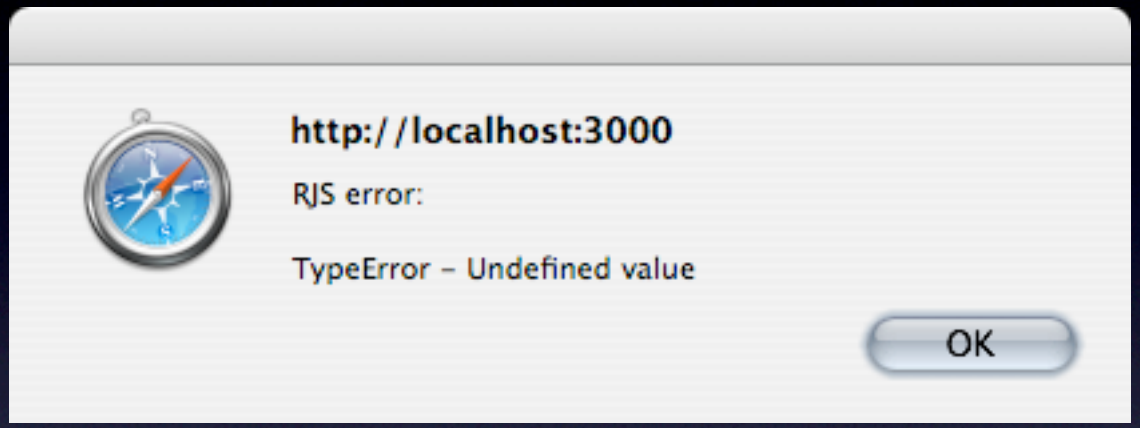
http://webkit.org/

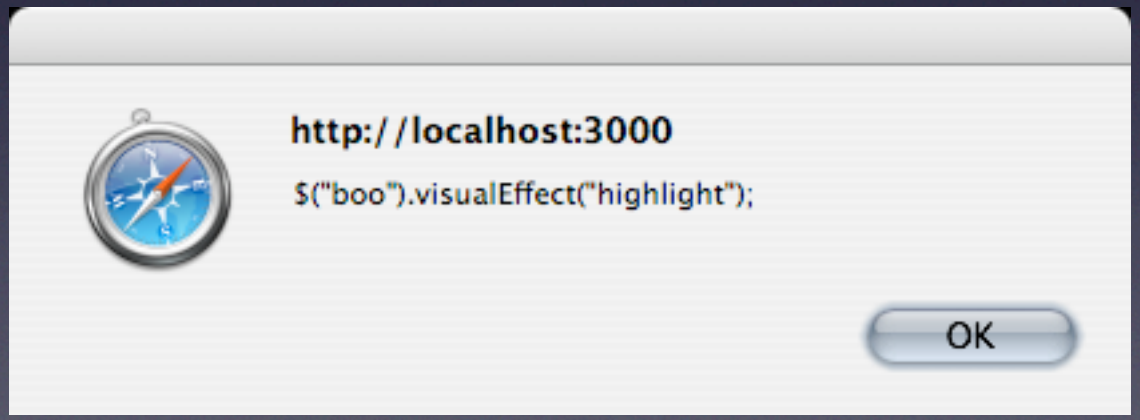
Microsoft Script Debugger for IE

http://blogs.msdn.com/ie/archive/2004/10/26/247912.aspx

Test with all browsers you want to support

```
config.action_view.debug_rjs = true
```

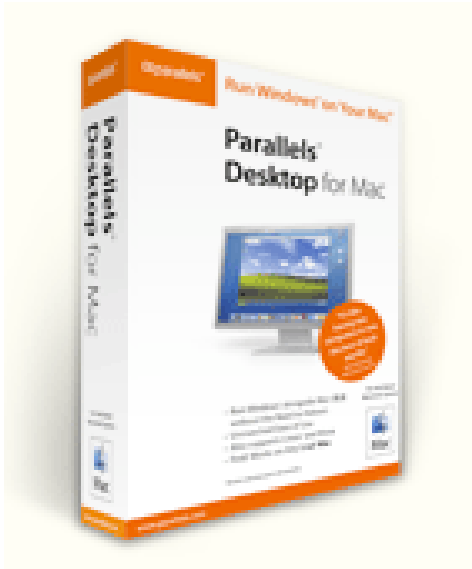




get a setup that
doesn‘t suck

Source: http://www.flickr.com/photos/icathing/26603225/

+


Parallels
Desktop for Mac

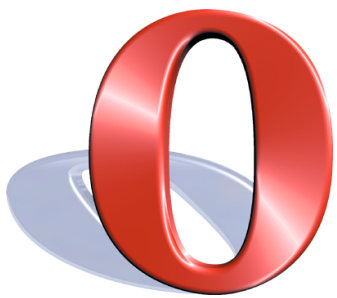
+


Microsoft
Windows xp
Professional

(+ ubuntu)


one machine to rule them all




help and ideas welcome

#prototype (freenode)
and
Rails Spinoffs Google Group