

JavaScript: Beyond the Basics

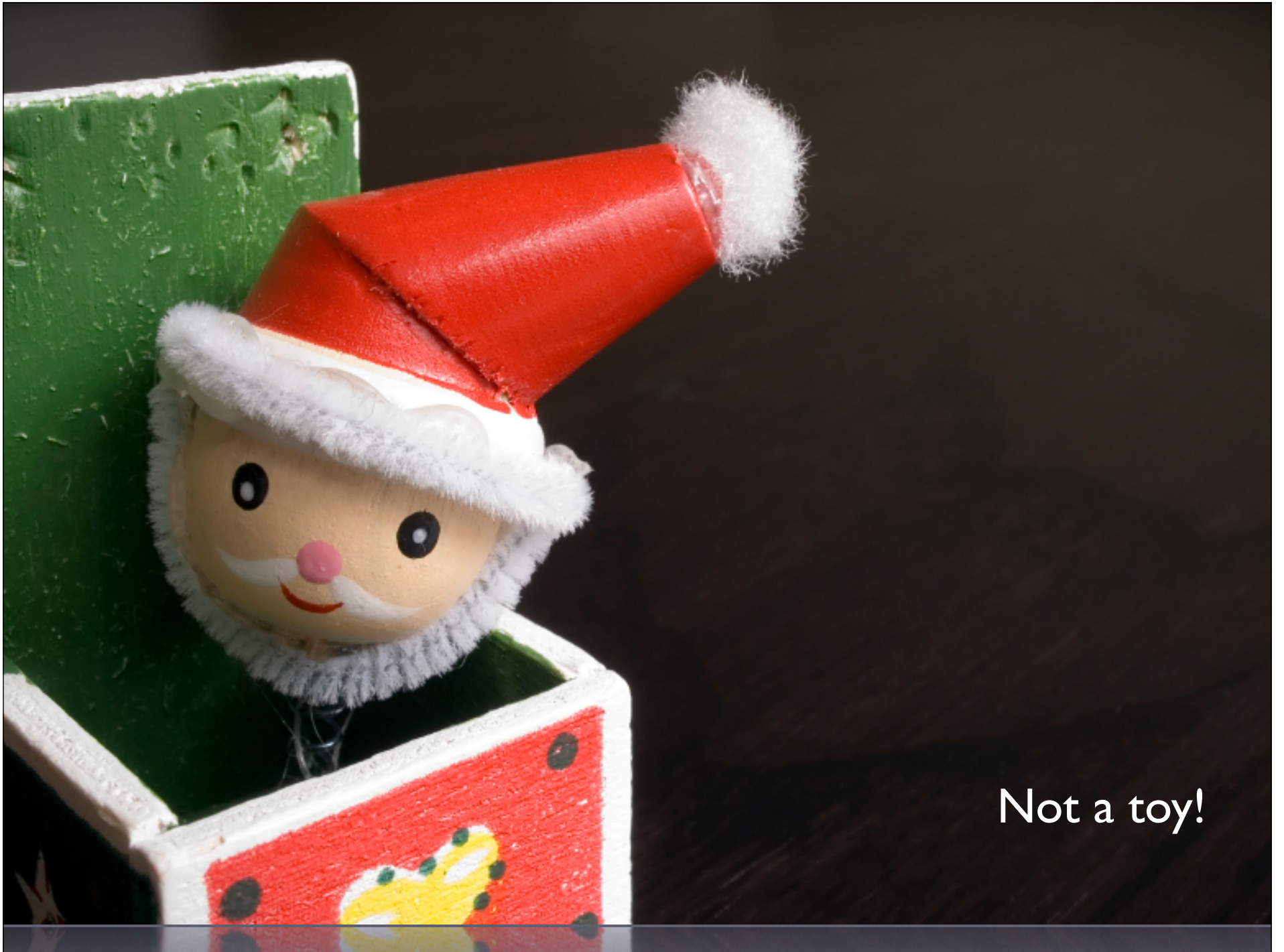
Nathaniel T. Schutta

Who am I?

- Nathaniel T. Schutta
<http://www.ntschutta.com/jat/>
- @ntschutta
- Foundations of Ajax & Pro Ajax and Java Frameworks
- UI guy
- Author, speaker, teacher
- More than a couple of web apps

The Plan

- Testing, testing, testing
- DSLs in JavaScript
- The Prototype
- Edge Cases
- Tools



Not a toy!

JavaScript has been
around for a while.

Many dismissed it as
“toy for designers.”

It's not the 90s anymore.

We have tools!

Developers care again!

Ajax.

Suffers from the “EJB issue.”

Powerful language.

Lisp with C syntax.

“The Next Big Language”

[http://steve-yegge.blogspot.com/
2007/02/next-big-language.html](http://steve-yegge.blogspot.com/2007/02/next-big-language.html)

Runs on lots of platforms
- including the JVM.

Ruby like?

“Rhino on Rails”

[http://steve-yegge.blogspot.com/
2007/06/rhino-on-rails.html](http://steve-yegge.blogspot.com/2007/06/rhino-on-rails.html)

Orto - JVM written in JavaScript.

<http://ejohn.org/blog/running-java-in-javascript/>

JS-909.

<http://www.themaninblue.com/experiment/JS-909/>

JSSpec.

JavaScript testing DSL.

```
/**  
 * Domain Specific Languages  
 */  
JSSpec.DSL = {};
```

BDD for JS.

Like RSpec.

Not quite as elegant.

```
describe('Plus operation', {
  'should concatenate two strings': function() {
    value_of("Hello " + "World").should_be("Hello World");
  },
  'should add two numbers': function() {
    value_of(2 + 2).should_be(4);
  }
})
```

value_of?

```
"Hello".should_be("Hello");
```

Sorry.

No method missing.

We'd need to modify
Object's prototype.

Generally a no-no.

Though it's been done.

<http://json.org/json.js>

Null, undefined objects.

Design choice - consistency.

```
describe('Plus operation', {
  'should concatenate two strings': function() {
    value_of("Hello " + "World").should_be("Hello World");
  },
  'should add two numbers': function() {
    value_of(2 + 2).should_be(4);
  }
})
```

describe - global
defined in JSSpec.js.

Creates a new
JSSpec.Spec()...

And adds it to an
array of specs.

value_of - global
defined in JSSpec.js.

value_of - converts parm
to JSSpec.DSL.Subject

Handles arbitrary objects.

JSSpec.DSL.Subject
contains should_*

Added to prototype.

```
JSSpec.DSL.Subject.prototype.should_be = function(expected) {  
  var matcher =  
    JSSpec.EqualityMatcher.createInstance(expected, this.target);  
  if(!matcher.matches()) {  
    JSSpec._assertionFailure = {message:matcher.explain()};  
    throw JSSpec._assertionFailure;  
  }  
}
```

this.target?

```
JSSpec.DSL.Subject = function(target) {  
  this.target = target;  
};
```

this in JS is...interesting.

Why is everything
JSSpec.Foo?

JS lacks packages
or namespaces.

Keeps it clean.

Doesn't collide...unless
you have JSSpec too!

Not just a DSL of course.

Defines a number
of matchers.

Also the runner
and the logger.

JSpec [X] Plus operation 2 examples 0 failures 0 errors 100% done 0.141 secs

JSpec home

Plus operation [rerun]

Plus operation [rerun]

should concatenate two strings

```
function () {  
  value_of("Hello World").should_be("Hello World");  
}
```

should add two numbers

```
function () {  
  value_of(4).should_be(4);  
}
```

Some CSS to make it pretty.

~ 1500 lines of code.

Clean code.

Why would you use it?

Easier to read.

```
function testStringConcat() {  
    assertEquals("Hello World", "Hello " + "World");  
}
```

```
function testNumericConcat() {  
    assertEquals(4, 2 + 2);  
}
```

```
var oTestCase = new YAHOO.tool.TestCase({
  name: "Plus operation",
  testStringConcat : function () {
    YAHOO.util.Assert.areEqual("Hello World", "Hello " + "World", "Should be 'Hello World'");
  },
  testNumericConcat : function () {
    YAHOO.util.Assert.areEqual(4, 2 + 2, "2 + 2 should be 4");
  }
});
```

```
describe('Plus operation', {
  'should concatenate two strings': function() {
    value_of("Hello " + "World").should_be("Hello World");
  },
  'should add two numbers': function() {
    value_of(2 + 2).should_be(4);
  }
})
```

Better? Worse?

What would you rather
read 6 months later?

<http://jania.pe.kr/aw/moin.cgi/JSSpec>

JsUnit.

Port of JUnit...to JS.

Runs in browser.

Can be automated.

Can be run on multiple
machines/OSs.

Simple HTML page.

Like a test class.

Create test methods.

Typical asserts
are supported.

Includes setUp/tearDown.

```
function testStringConcat() {  
    assertEquals("Hello World", "Hello " + "World");  
}  
function testNumericConcat() {  
    assertEquals(4, 2 + 2);  
}
```

Test runner.

Can run it on web server.

Includes set of ant tasks.

<http://www.jsunit.net/>

YUI Test.

Test framework from YUI.

Not a direct xUnit port.

Test cases.

```
var oTestCase = new YAHOO.tool.TestCase({
  name: "Plus operation",
  testStringConcat : function () {
    YAHOO.util.Assert.areEqual("Hello World", "Hello " + "World", "Should be 'Hello World'");
  },
  testNumericConcat : function () {
    YAHOO.util.Assert.areEqual(4, 2 + 2, "2 + 2 should be 4");
  }
});
```

setUp & tearDown.

Ability to ignore tests.

`_should.error` property.

fail().

Asserts.

Sameness assertions.

Data type assertions.

isFalse(), isTrue(), isNaN(),
isNull(), isNotNull(),
isUndefined()...

DOM simulation.

Not everything.

Clicks, mouse moves,
key events.

Asynchronous capability.

Allows tests to wait.

`wait()` & `resume()` -
similar to `setTimeout()`.

Also includes test suites.

Includes suite level
setUp/tearDown.

Basic test runner.

Not very pretty ;)

Other visualizations likely.

Quite new.

Very intriguing...

<http://developer.yahoo.com/yui/yuitest/>

Blue Ridge.

Rails plugin.

Fast, headless.

Works with a variety of testing frameworks.

Mocking support.

<http://github.com/relevance/blue-ridge>

<http://blog.thinkrelevance.com/2009/5/12/blue-ridge-1-0-javascript-unit-testing-for-rails-scandalous>

JSLint.

JS verifier.

Doug Crockford.

Not nice...

Web based.

Can paste code in.

Also runs on Rhino.

<http://www.jshint.com/rhino/index.html>

<http://www.jshint.com/>

DSLs in JS.

Coffee!

Viabile option.

Widely used language.

Not necessarily easy.

Syntax isn't as flexible.

Lots of reserved words.

Freakn ;

Hello prototype!

Verbs as first class citizens.

[http://steve-yegge.blogspot.com/2006/03/
execution-in-kingdom-of-nouns.html](http://steve-yegge.blogspot.com/2006/03/execution-in-kingdom-of-nouns.html)

Object literals.

DSL vs. named parameters
vs. constructor parms.

```
new Ajax.Request('/DesigningForAjax/validate', {
  asynchronous: true,
  method: "get",
  parameters: {zip: $F('zip'), city: $F('city'), state: $F('state')},
  onComplete: function(request) {
    showResults(request.responseText);
  }
})
```

Fail fast vs. fail silent.

Method chaining is trivial.

Context can be a challenge.

Documentation key.

PDoc, YUI Doc.

https://www.ohloh.net/p/pdoc_org

<http://developer.yahoo.com/yui/yuidoc/>

Prototypal Inheritance.

JavaScript doesn't
have classes.

Objects are king.

Reuse via cloning.

Every object has a
prototype.

Just a link.

Set in the constructor.

Calls move “up the chain.”

Can change the prototype!

Can add things to
the prototype.

```
Foo.prototype.bar = function() { //do bar};
```

It's not that hard!

Just not “classical.”

You can simulate
classical inheritance.

<http://api.prototypejs.org/language/class.html>

<http://javascript.crockford.com/inheritance.html>

Want more?

<http://javascript.crockford.com/prototypal.html>

Many language use
prototypical inheritance.

loke.

<http://ioke.org/>

Edge Cases.

Semicolon insertion.

Semicolons are optional...

New line? Insert it!

Works...until it doesn't.

```
return
```

```
    true
```

```
return;  
    true;
```

Don't rely on it!

Reserved words!

JS has too many.

The odd case of `typeof`.

Returns a string.

Generally right...

`typeof Array...object`
`typeof null...object`

Truthy.

Remember C?

Coercion!

True becomes a 1
False becomes a 0.

Number becomes true
0 or NaN becomes false.

Strings are true
empty strings are false.

Null and undefined are false
Non-null {}, [], function, true.

Two scopes:
function and global.

Syntax is reused.

Objects...like arrays.

Collections, with keys
and values...hash!

Constructors can make
private variables!

```
function Foo() {  
    var mine = 1;  
}
```

Tools.

First rule of JavaScript:

Use Firefox.

Second rule of JavaScript:

Use Firefox.

JavaScript console.

Web developer tools.

Firebug!



<http://www.joehewitt.com/about.php>

Firebug - Google

Console HTML CSS Script DOM Net Hammerhead YSlow

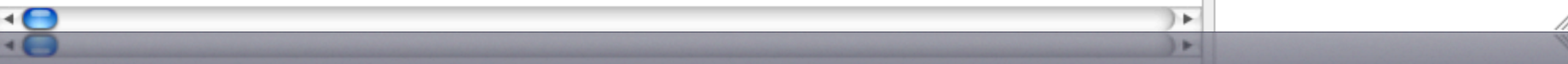
all www.google.com

```
1 <!doctype html><html onmousemove="google&&google.fade&&google.fade()"><head><meta http-equiv="content-
2 (function(){for(var d=0,c=["ad","bc","p","pa","zd","ac","pc","pah","ph","sa","xx","zc","zz"][d++]);
3 function(){google.j.pl.push([b,arguments])})(c)})(c)});
4 window.google.sn="webhp";window.google.timers={load:{t:{start:(new Date).getTime()}}};try{window.goog
5 </script><style>td{line-height:.8em;}.gac_m td{line-height:17px;}form{margin-bottom:20px;}body,td,a,p
6 google.y={};google.x=function(e,g){google.y[e.id]=[e,g];return false};if(!window.google)window.google
7 return true};
8 window.gbar={qs:function(){},tg:function(e){var o={id:'gbar'};for(i in e)o[i]=e[i];google.x(o,function
9 google.x.js=1;google.y.first.push(function(){google.ac.m=1;google.ac.b=true;google.ac.i(document.f,do
10 var a=window.google.f={};a.f=1;a.s=1;a.a=(new Date).getTime();google.rein.push(function(){a.f=1;a.s=1
11 document.getElementsByTagName(b),d=0,c=f[d];c=f[d++]);if(c.className==h)e.push(c);return e}google.fade
12 var i=["input","lsb"];g.push(i);var j=m(i[0],i[1]);function n(){j[0].style.filter=(j[1].style.filter=
13 })(c)});
14 });google.x.js&&google.j&&google.j.xi&&google.j.xi()</script></div><script>(function(){
15 function a(){google.timers.load.t.ol=(new Date).getTime();google.report&&google.report(google.timers.
16 })();
17 </script>
```

Search Off

Watch Stack Breakpoints

New watch expression...



```
13 </script>
14 })();
```

JavaScript debugging!

Console - real errors.

Log to the console.

CSS/HTML inspector.

Modify CSS.

DOM explorer.

Works in FF but not IE?

Firebug Lite.

<http://getfirebug.com/lite.html>

YSlow.

<http://www.getfirebug.com/>
<http://developer.yahoo.com/yslow/>

JSLint.

VERY strict.

<http://www.JSLint.com/>

Will it run on IE/FF/Opera?

Cross Check.

Doesn't require you to
install all the browsers!

<http://www.thefrontside.net/crosscheck>

Google - Closure Tools.

<http://code.google.com/closure/>

A “compiler.”

Analyzes and minimizes.

Command line, web app
or RESTful API.

Also includes a JS library
and a templating system.

IDE support is
getting better.

JavaScript isn't a toy.

Not quite as flexible as
some languages.

Plenty of metaprogramming
goodness!

Beware the edges!

Questions?!?

Thanks!

Please complete your surveys.