DSLs in JavaScript

Nathaniel T. Schutta

Who am I?

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

The Plan

- DSLs?
- JavaScript? Seriously?
- Examples
- Lessons Learned

DS what now?

Domain Specific Language.

Every domain has its own language.

“Part of the benefit of being "into" something is having an insider lexicon.”

Kathy Sierra
Creating Passionate Users

http://headrush.typepad.com/creating_passionate_users/2006/11/why_web_20_is_m.html

Three quarter, knock down, soft cut.

Scattered, smothered, covered.

Large skim mocha, no whip no froth.

Not general purpose.

Simpler, more limited.

Expressive.

Terse.

```
$$('.header').each(function(el) {el.observe("click", toggleSection)});
```

Not a new idea.

Unix.

Little languages.

Lisp.

Build the language up...

Lots of attention today.

Rails!

Ruby is very hospitable.

So are other languages ;)

Internal vs. External.

Internal.

Within an existing language.

More approachable.

Simpler.

No grammars, parsing, etc.

Constrained by host language.

Flexible syntax helps!

Ruby ;)

Fluent interface.

Embedded DSLs.

External.

Create your own language.

Grammars.

Need to parse.

ANTLR, yacc, JavaCC.

Harder.

More flexibility.

Language workbenches.

Tools for creating new languages.

Internal are more common today.

Language workbenches - shift afoot?

http://martinfowler.com/articles/mpsAgree.html

http://martinfowler.com/articles/languageWorkbench.html

Meta Programming System.

http://www.jetbrains.com/mps/

Intentional Programming - Charles Simonyi.

http://intentsoft.com/

http://www.technologyreview.com/Infotech/18047/?a=f

Oslo.

http://msdn.microsoft.com/en-us/oslo/default.aspx

Xtext.

http://wiki.eclipse.org/Xtext

Why are we seeing DSLs?

Easier to read.

Closer to the business.

Less friction, fewer translations.

Biz can review...

“Yesterday, I did a code review. With a CEO... Together, we found three improvements, and a couple of outright bugs.”

Bruce Tate
Canaries in the Coal Mine

http://blog.rapidred.com/articles/2006/08/30/canaries-in-the-coal-mine

Don’t expect them to write it though!

Will we all write DSLs?

No.

Doesn't mean we can't use them.

General advice on building a DSL:

Write it as you'd like it to be...

Even on a napkin!

Use valid syntax.

Iterate, iterate, iterate.

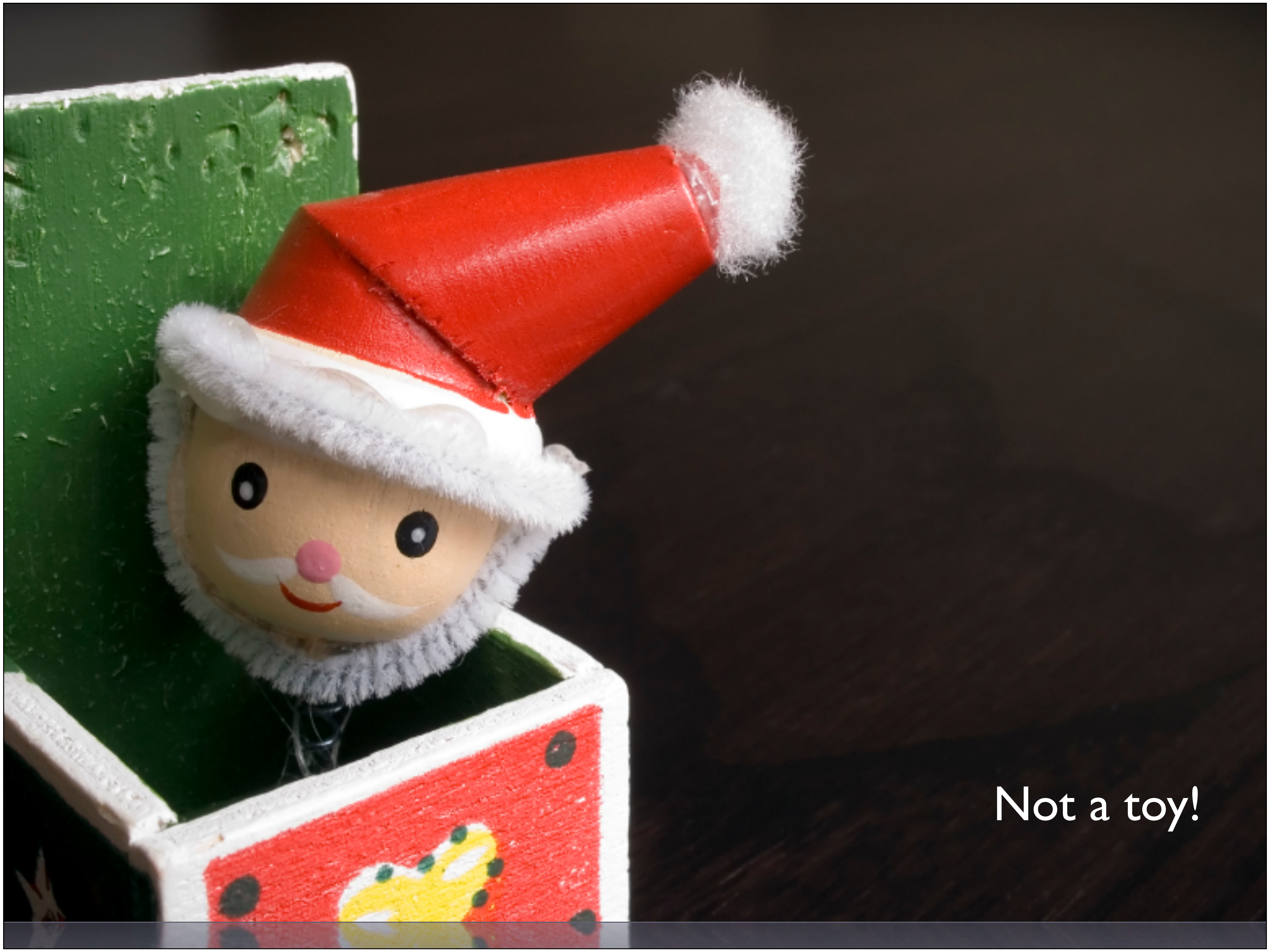
Work on the implementation.

http://martinfowler.com/dslwip/

http://weblog.jamisbuck.org/2006/4/20/writing-domain-specific-languages

http://memeagora.blogspot.com/2007/11/ruby-matters-frameworks-dsls-and.html

http://martinfowler.com/bliki/DslQandA.html


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.

“The Next Big Language”

http://steve-yegge.blogspot.com/
2007/02/next-big-language.html

Runs on lots of platforms - including the JVM.

Ruby like?

“Rhino on Rails”

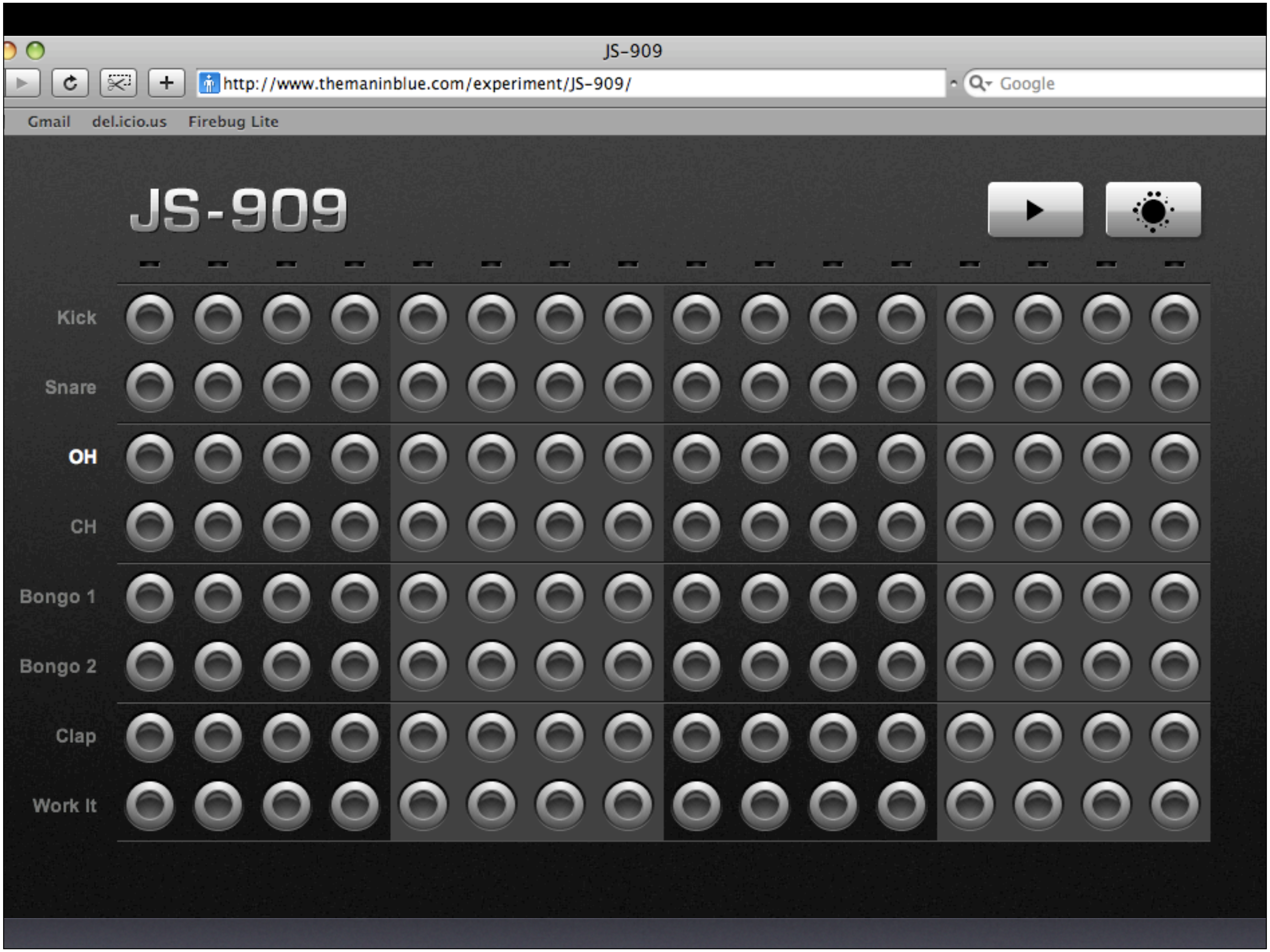
http://steve-yegge.blogspot.com/2007/06/rhino-on-rails.html

Orto - JVM written in JavaScript.

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

JS-909.

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


JS-909
http://www.themaninblue.com/experiment/JS-909/
Google
Gmail
del.icio.us
Firebug Lite
JS-909
Kick
Snare
OH
CH
Bongo 1
Bongo 2
Clap
Work It

JSSpec.

JavaScript testing DSL.

JSSpec? Really?

```
/**
 * 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.

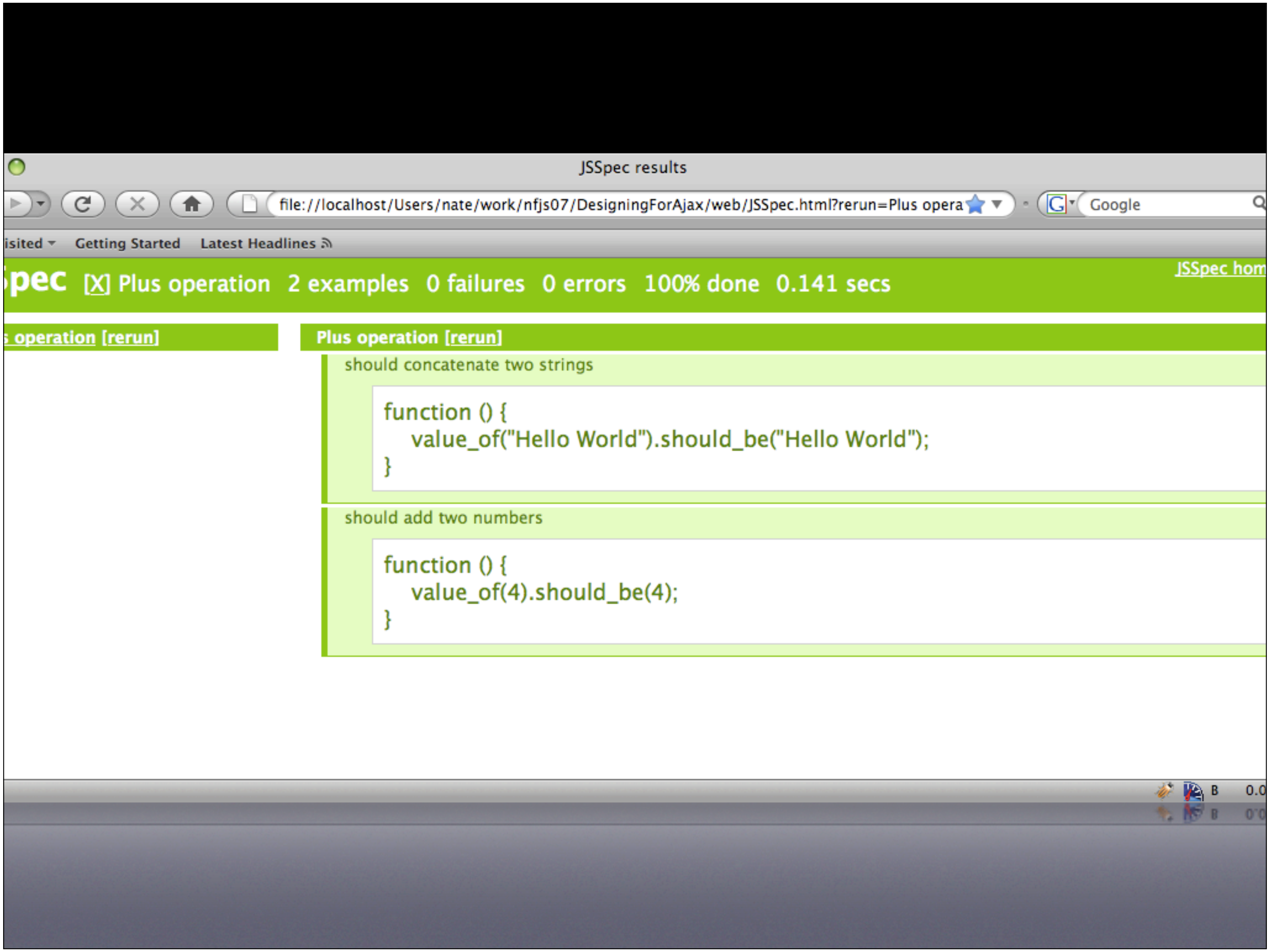
Keeps it clean.

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.


JSSpec results
file://localhost/Users/nate/work/nfjs07/DesigningForAjax/web/JSSpec.html?rerun=Plus opera
Google
isited
Getting Started
Latest Headlines
JSSpec hom

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

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

ActiveRecord.js

JavaScript ORM.

Seriously?

Yep.

Let’s you use a DB from JavaScript.

Client or server ;)

Gears, AIR, W3C
HTML5 SQL spec.

In-memory option too.

Some free finder methods.

Base find method.

Migrations.

```
ActiveRecord.Migrations.migrations = {
  1: {
    up: function(schema){
      schema.createTable('one',{
        a: '',
        b: {
          type: 'TEXT',
          value: 'default'
        } });
    },
    down: function(schema){
      schema.dropTable('one');
    }
  }
};
```

Validations.

```
User.validatesPresenceOf('password');
```

More to come...

Supports basic relationships.

Early stages...

On GitHub, contribute!

http://activerecordjs.org/

Objective-J

Objective-C...for JS.

JavaScript superset.

Cheating...kind of.

Native and Objective-J classes.

Allows for instance methods.

Parameters are separated by colons.

```
- (void)setJobTitle: (CPString)aJobTitle
  company: (CPString)aCompany
```

Bracket notation for method calls.

```
[myPerson setJobTitle: "Founder" company: "280 North"];
```

Does allow for method_missing.

http://cappuccino.org/discuss/2008/12/08/on-leaky-abstractions-and-objective-j/

http://cappuccino.org/

Coffee DSL.

Lessons learned.

Viable option.

Widely used language.

Not necessarily easy.

Syntax isn’t as flexible.

Lots of reserved words.

Freakn ;

Hello prototype!

Verbs as first class citizens.

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/

JavaScript isn’t a toy.

Not quite as flexible.

Plenty of metaprogramming goodness!

Questions?!?

Thanks!

Please complete your surveys.