refactoring javascript

stuart halloway
http://thinkrelevance.com

ground rules

cover code with tests

don’t code javascript naked

do the “traditional” oo refactorings

also do functional refactorings

unit testing:

screw.unit

screw.unit example

```
Screw.Unit(function(){
  describe("Your application javascript", function(){
    it("does something", function(){
      expect("hello").to(equal, "hello");
    });
  });
});
```

mocking:

smoke

smoke example

```
it("can stub with Smoke!", function() {
  stub(Foo, "bar").and_return(7);
  expect(Foo.bar()).to(equal, 7);
});

it("can mock with Smoke!", function() {
  mock(Foo).should_receive("bar")
    .with_arguments(10).exactly(1, "time").and_return(42);
  expect(Foo.bar(10)).to(equal, 42);
});
```

javascript attire:

jquery

putting it all together: **blue-ridge**

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

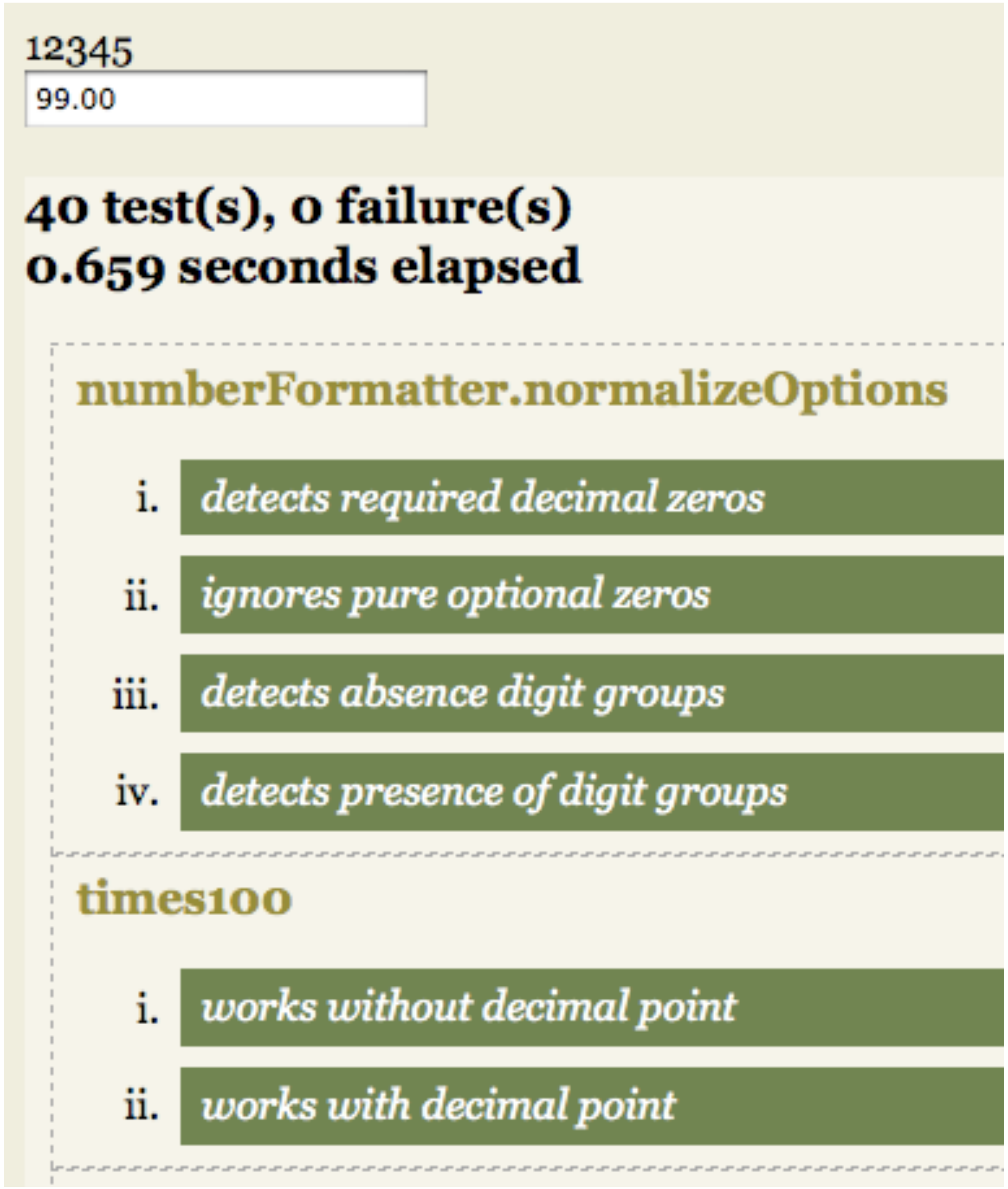
headless builds

```
rake test:javascripts
(in /Users/stuart/presentations/refactoring-javascript)
Running application_spec.js with fixture 'fixtures/application.html'...
..

2 test(s), 0 failure(s)
0.456 seconds elapsed
Running numberformatter_spec.js with fixture 'fixtures/numberformatter.html'...
........................................

40 test(s), 0 failure(s)
0.518 seconds elapsed
```

in-browser builds



12345

99.00

40 test(s), 0 failure(s)
0.659 seconds elapsed

numberFormatter.normalizeOptions

i. *detects required decimal zeros*
ii. *ignores pure optional zeros*
iii. *detects absence digit groups*
iv. *detects presence of digit groups*

times100

i. *works without decimal point*
ii. *works with decimal point*

something to refactor

http://code.google.com/p/jquery-numberformatter/

↓

http://github.com/stuarthalloway/refactoring-number-formatter

covering tests document what you have

need a fixture

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
                      "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>Numberformatter | JavaScript Testing Results</title>
  <link rel="stylesheet" href="screw.css" type="text/css" charset="utf-8" />
  <script type="text/javascript"
          src="../../../vendor/plugins/blue-ridge/lib/blue-ridge.js"></script>
</head>

<body>
  <div id="value"></div>
  <form>
    <input id="input" type="text"></input>
  </form>
</body>

</html>
```

node to
operate on

default to u.s. format

```
it("defaults to us #,###.00", function(){
  $("#value").text(1999);
  $("#value").format();
  expect($("#value").text()).to(equal, "1,999.00");
});
```

percents

```
it("supports percents", function(){
  $("#value").text(".25");
  $("#value").format({format: "##%"});
  expect($("#value").text()).to(equal, "25%");
});
```

input elements

```
it("works with input elements", function(){
  $("#input").val(99);
  $("#input").format();
  expect($("#input").val()).to(equal, "99.00");
});
```

non-format characters

```
it("ignores non-format characters at start and end",

function(){
  $("#value").text("42");
  $("#value").format({format: "BOO ## YAA"});
  expect($("#value").text()).to(equal, "BOO 42 YAA");
});
```

negative prefix

```
it("handles negative prefix, then non-format
characters then number, then non-format",

function(){
  $("#value").text("-500,000.77");
  $("#value").format({format: "-$#.#"});
  expect($("#value").text()).to(equal, "-$500000.8");
});
```

forcing decimal

```
it("shows decimal for whole numbers if forced",

function(){
  $("#value").text("15");
  $("#value").format({
    format: "#.##",
    decimalSeparatorAlwaysShown: true
  });
  expect($("#value").text()).to(equal, "15.");
});
```

refactoring #1: extract method

parsing options string

```
function parseOptionsFormat(options) {
  var validFormat = "0#-,.";

  // strip all the invalid characters at the beginning and the end
  // of the format, and we'll stick them back on at the end
  // make a special case for the negative sign "-" though, so
  // we can have formats like -$23.32
  options.prefix = "";
  options.negativeInFront = false;
  for (var i=0; i<options.format.length; i++)
  {
     if (validFormat.indexOf(options.format.charAt(i))==-1)
         options.prefix = options.prefix + options.format.charAt(i);
     else if (i==0 && options.format.charAt(i)=='-')
     {
        options.negativeInFront = true;
        continue;
     }
     else
         break;
  }
  options.suffix = "";
  for (var i=options.format.length-1; i>=0; i--)
  {
     if (validFormat.indexOf(options.format.charAt(i))==-1)
         options.suffix = options.format.charAt(i) + options.suffix;
     else
         break;
  }

  options.format = options.format.substring(options.prefix.length);
  options.format = options.format.substring(0, options.format.length - options.suffix.length);
};
```

our enemies

control flow

interrupted control flow

variables

refactoring #2:
use the right tools

use regular expressions

```
function parseOptionsFormat(options) {
  var match = /^(-?)([^-0#,.]*)([-0#,.]*)([^-0#,.]*)$/.exec(options.format);
  if (!match) throw "invalid number format " + options.format;
  options.negativeInFront = (match[1] == "-");
  options.prefix = match[2];
  options.format = match[3];
  options.suffix = match[4];
};
```

testing exceptions

```
it("throws up if it finds non-format characters in
the middle",

function(){
  $("#value").text("767");
  expect(function () {
    $("#value").format({
      format: "## AND ##"
    })
  }).to(throw_object,
         "invalid number format ## AND ##");
});
```

extending Screw.Unit

```
// TODO: add to Screw.Unit
throw_object: {
  match: function(object, actual_fn) {
    actual_fn._last_err = "[no error]";
    try {
      actual_fn();
      return false;
    } catch (e) {
      actual_fn._last_err = e;
      return e === object;
    }
  },

  failure_message: function(expected_exc, actual_fn, not) {
    return 'expected ' + $.print(actual_fn) + (not ? ' to not ' : '
to ') + 'throw ' + $.print(expected_exc) + ' not "' +
actual_fn._last_err + '"';
  }
},
```

refactoring #3
extract method

```
if (jQuery(this).is(":input"))
   jQuery(this).val(returnString);
else
   jQuery(this).text(returnString);
```

```
jQuery.fn.valOrText = function() {
  return (
    jQuery(this).is(":input") ?
      jQuery.fn.val : jQuery.fn.text
  ).apply(this,arguments);
};
```

refactoring #4: imperative loop => declarative re

```
while (text.indexOf(group)>-1)
       text = text.replace(group,'');
var number = new Number(text.replace(dec,".").replace(neg,"-"));
```

```
// technical debt: what happens to numbers with more than one ‘
// decimal or negative sign?
var number = new Number(text.replace(new RegExp(group, "g"), "")
                            .replace(dec,".")
                            .replace(neg,"-"));
```

refactoring #5: kill dead code

anybody using this?

```
jQuery.formatNumber = function(number, options) {
    var options = 
jQuery.extend({},jQuery.fn.parse.defaults, options);
    var formatData = 
formatCodes(options.locale.toLowerCase());

    var dec = formatData.dec;
    var group = formatData.group;
    var neg = formatData.neg;

    var numString = new String(number);
    numString = 
numString.replace(".",dec).replace("-",neg);
    return numString;
};
```

breaking change #1:

23z4 => 23, not 234

recognize numbers

```
it("knows all the valid number characters",

function(){
  $("#value").text("-123,456.789");
  expect(
    $("#value").parse()).to(equal, [-123456.789]
  );
});
```

ignore trailing junk

```
it("ignores junk at the end", function(){
  $("#value").text("36XL");
  expect($("#value").parse()[0]).to(equal, 36);
});
```

ignore trailing digits

```
it("ignores everything after the first non-number
character",

function(){
  $("#value").text("14 to 16");
  expect($("#value").parse()[0]).to(equal, 14);
});
```

breaking change #2:

big numbers

zero format digits

```
it("handles zero format digits", function() {
  expect($.numberFormatter.formatNumber(
    "123.45",
    {decimalsRightOfZero: 0}
  )).to(equal, "123");
});
```

a few format digits

```
it("handles a few format digits", function() {
  expect($.numberFormatter.formatNumber(
    "0.0136",
    {decimalsRightOfZero: 2}
  )).to(equal, "0.01");
});
```

many format digits

```
it("handles a lot of format digits", function() {
  expect($.numberFormatter.formatNumber(
    "1.01234567890001",
    {decimalsRightOfZero: 14}
  )).to(equal, "1.01234567890001");
});
```

format > actual

```
it("handles more format digits than actual digits",

function() {
  expect($.numberFormatter.formatNumber(
    "1.5",
    {decimalsRightOfZero: 8}
  )).to(equal, "1.50000000");
});
```

rounding

```
it("handles more format digits than actual digits",

function() {
  expect($.numberFormatter.formatNumber(
    "1.5",
    {decimalsRightOfZero: 8}
  )).to(equal, "1.50000000");
});
```

opportunities or risks?

corner cases

range limitations

exceptional conditions

generalizations

specializations

refactoring #5: convert classes and functions to data

covering tests

```
describe("formatCodes", function() {
  it("knows us and friends", function() {
    expect($.numberFormatter.formatCodes("en")).to(
      equal,  { dec: ".", group: ",", neg: "-" });
  });

  it("knows es and friends", function() {
    expect($.numberFormatter.formatCodes("es")).to(
      equal,  { dec: ",", group: ".", neg: "-" });
  });

  // etc.
});
```

data hidden in code

```
function formatCodes(locale) {

     // default values
     var dec = ".";
     var group = ",";
     var neg = "-";

     if (locale == "us" ||
         locale == "in"
        )
     {
          dec = ".";
          group = ",";
     }

    // ... dozens of lines elided ...
    return new FormatData(dec, group, neg);

};
```

```
function FormatData
(dec, group, neg) {
   this.dec = dec;
   this.group = group;
   this.neg = neg;
};
```

data, revealed

```
var us_et_al = { dec: ".", group: ",", neg: "-" }
var eu_et_al = { dec: ",", group: ".", neg: "-" }
var cz_et_al = { dec: ",", group: " ", neg: "-" }
nf.formatCodesTable = {
  us: us_et_al,
  ae: us_et_al,
  de: eu_et_al,
  es: eu_et_al,
  cz: cz_et_al,
  fr: cz_et_al,
  ch: {group: "'", dec: ".", neg: "-"}
};

nf.formatCodes = function(s) {
 return nf.formatCodesTable[s] || nf.formatCodesTable["us"];
};
```

...and some debt

```
// Technical debt: would like to see this throw error
it("brooks no nonsense", function() {
  expect($.numberFormatter.formatCodes("splat")).to(
    equal,  { dec: ".", group: ",", neg: "-" });
});
```

recap

start with covering tests

don’t code naked

functions longer than seven lines are bad

always prefer data

mark breaking changes/tech debt

Code and Slides:

http://github.com/stuarthalloway/refactoring-number-formatter

Email:	stu@thinkrelevance.com
Office:	919-442-3030
Twitter:	twitter.com/stuarthalloway
Facebook:	stuart.halloway
Github:	stuarthalloway

Talks:	http://blog.thinkrelevance.com/talks
Blog:	http://blog.thinkrelevance.com
Book:	http://tinyurl.com/clojure