TDD WITH MOCHA

THE GOAL OF TDD

"Clean code that works"

Kent Beck, "Test-Driven Development by Example"

TDD IN FOUR STEPS

1. Write a test
2. Run tests and see the new test fail
3. Write code to pass the test
4. Refactor

A feature-rich JavaScript testing framework designed for asynchronous testing.

A FIRST TASTE OF MOCHA

01_getting_started.js:

```
/*global describe:false, it:false*/
'use strict';

var assert = require('chai').assert;

describe('Array', function () {
  describe('#indexOf()', function () {

    it('should return -1 when the value is not present', function () {
      assert.equal(-1, [1, 2, 3].indexOf(5));
      assert.equal(-1, [1, 2, 3].indexOf(0));
    });

  });
});
```

SETTING UP YOUR ENVIRONMENT

```
$ npm init                              # set up your project
                                        # set "test command" to: mocha test
$ npm install mocha chai --save-dev     # install and save dependencies
$ mkdir test                            # a directory to put your test file
```

package.json should look like something like this:

```
{
  "name": "src",
  "version": "0.0.0",
  "main": "index.js",
  "scripts": {
    "test": "mocha test/*.js -R spec"
  }
}
```

Put a test in the *test* directory and execute the tests:

```
$ cp 01_getting_started.js test/
$ npm test
```

```
Array
  #indexOf()
    ✓ should return -1 when the value is not present


1 passing (4ms)
```

CHAI ASSERTION LIBRARY

- Mocha can use any assertion library, even core "assert", they simply need to throw exceptions
- Chai provides BDD (should/expect), or traditional TDD assertions
- Node assertions, are generally the reverse of traditional xUnit assertions:

```
assert.equal(actual, expected, message); // *actual* before *expected*
```

CHAI ASSERTIONS

Assertions all have the equivalent **not** versions available for the inverse

```
// include just the TDD-style assertions
var assert = require('chai').assert;

assert(expression, message);                        // assert "truthy" exp
assert.ok(object, message);                         // assert "truthy" obj

assert.equal(actual, expected, message);            // assert non-strict =
assert.strictEqual(actual, expected, message);      // assert strict === e
assert.deepEqual(actual, expected, message);        // assert deep object

assert.typeOf(value, name, message);                // assert `typeof valu
assert.instanceOf(object, constructor, message);    // object is instance

// throw an exception displaying actual and expected
assert.fail(actual, expected, message);

isTrue, isFalse, isNull, isUndefined, isDefined,
isFunction, isObject, isArray, isString, isNumber, isBoolean,
```

... and more, see http://chaijs.com/api/assert/

ASYNCHRONOUS TESTING

02_async_test.js:

```
/*global describe:false, it:false*/
'use strict';

var assert = require('chai').assert,
    fs = require('fs');

describe('Core fs', function () {
  describe('stat()', function () {

    it('should return an error when file does not exist', function (next

      fs.stat('/path/to/nonsense/file', function (err, stat) {
        assert.ok(err, 'received an error');
        assert.equal(err.code, 'ENOENT', 'received correct error type');
        assert.notOk(stat, 'did not receive a "stat"');

        //next();
      });

    });

  });
});
```

```
$ cp 02_async_test.js test
$ npm test
```

FIXTURE SET-UP AND TEAR-DOWN

Let's write a simple test for `fs.chmod()`

What set-up do we need?

- A file to change mode of
- The file needs to start with one mode so we can change it to a different mode

How do we execute the test?

- Change the access permissions (mode) of a file and confirm that the new mode has stuck

What tear-down do we need?

- Remove the test file

FIXTURE SET-UP AND TEAR-DOWN

03_fixtures.js:

```
/// fs.chmod() test executor
function testModeChange (mode, next) {
  fs.stat(testfile, function (err, stat) { // sanity check, original
    assert.notOk(err);
    assert.equal('644', stat.mode.toString(8).substring(3), 'correct

    fs.chmod(testfile, mode, function (err) { // execute tested meth
      assert.notOk(err, 'no error');

      fs.stat(testfile, function (err, stat) { // check mode has cha
        assert.notOk(err);
        assert.equal(mode, stat.mode.toString(8).substring(3), 'corr
        next();
      });

    });

  });
}

it('should change mode to 755', function (next) {
  testModeChange('755', next);
});
it('should change mode to 400', function (next) {
  testModeChange('755', next);
});
it('should change mode to 222', function (next) {
  testModeChange('222', next);
});
```

FIXTURE SET-UP AND TEAR-DOWN

03_fixtures.js:

```
/*global describe:false, it:false, before:true, after:true, beforeEach:t
'use strict';

var assert = require('chai').assert,
    os = require('os'),
    fs = require('fs'),
    path = require('path');

describe('Core fs', function () {
  describe('chmod()', function () {

    var testfile = path.join(os.tmpDir(), 'test_core_fs.' + process.pid)

    before(function (next) {
      fs.writeFile(testfile, 'contents', next); // dummy file with dummy
    });

    after(function (next) {
      fs.unlink(testfile, next); // delete
    });

    beforeEach(function (next) {
      fs.chmod(testfile, '644', next); // reset file permissions to 644
    });

    /// fs.chmod() test executor

  });
});
```

FIXTURE SET-UP AND TEAR-DOWN

```
$ cp 03_fixtures.js test/
$ npm test
```

```
Core fs
  chmod()
    ✓ should change mode to 755
    ✓ should change mode to 400
    ✓ should change mode to 222


3 passing (6ms)
```

CODE COVERAGE WITH ISTANBUL

Something to test: **Quantity** is an object to hold and convert physical quantities

```
var oneLb = new Quantity(1, Quantity.LB);

var asKg = oneLb.convertTo(Quantity.KG);

console.log('1 lb =', asKg.value, 'kg'); // 1 lb = 0.45359237 kg
```

Full implementation at https://github.com/rvagg/quantities

CODE COVERAGE WITH ISTANBUL

04_quantities.js

```
function Quantity (value, units) {
  this.value = value;
  this.units = units;
}

Quantity.MM = 'mm';
Quantity.M = 'm';
Quantity.IN = 'inches';
Quantity.KG = 'kg';
Quantity.LB = 'lb';

var toInternalUnits = { };
toInternalUnits[Quantity.LB] = 45359237 / 100000000;
toInternalUnits[Quantity.KG] = 1;
toInternalUnits[Quantity.MM] = 0.001;
toInternalUnits[Quantity.M] = 1;
toInternalUnits[Quantity.IN] = 25.4 / 1000;

Quantity.prototype.convertTo = function (units) {
  if (this.units == units)
    return this;
  if (!toInternalUnits[this.units] || !toInternalUnits[units])
    throw new TypeError('Unknown units: ' + this.units + ' to ' + units)
  if (typeof this._internalUnits != 'number')
    this._internalUnits = this.value * toInternalUnits[this.units];
  return new Quantity(this._internalUnits * (1 / toInternalUnits[units])
};

module.exports = Quantity;
```

CODE COVERAGE WITH ISTANBUL

05_quantities_test.js

```
/// Test Quantity object
describe('Quantity', function () {

  it('should hold value and units', function () {
    var quantity = new Quantity(101.101, Quantity.KG);
    assert.equal(quantity.value, 101.101, 'correct value');
    assert.equal(quantity.units, Quantity.KG, 'correct units');
  });

  describe('conversions', function () {

    function testConversion (srcQuantity, dstValue, dstUnits) {
      var quantity = srcQuantity.convertTo(dstUnits);
      assert.equal(quantity.units, dstUnits, 'converted has ' + dstUnits
      assert(Math.abs(quantity.value - dstValue) < 0.001, 'converted to
    }

    it('should convert from kilograms to pounds', function () {
      testConversion(new Quantity(101, Quantity.KG), 222.667, Quantity.L
    });
    it('should convert from pounds to kilograms', function () {
      testConversion(new Quantity(222.667, Quantity.LB), 101, Quantity.K
    });

  });

});
```

CODE COVERAGE WITH ISTANBUL

Set up Istanbul and wrap Mocha:

```
$ npm install istanbul --save-dev
```

Edit *package.json* and update "tests" script:

```
  "scripts": {
    "test": "istanbul cover _mocha -- test/*.js -R spec"
  }
```

Test output now includes code coverage information.

Prepare examples and run tests:

```
$ rm test/*
$ cp 05_quantities_test.js test/
$ npm test
```

CODE COVERAGE WITH ISTANBUL

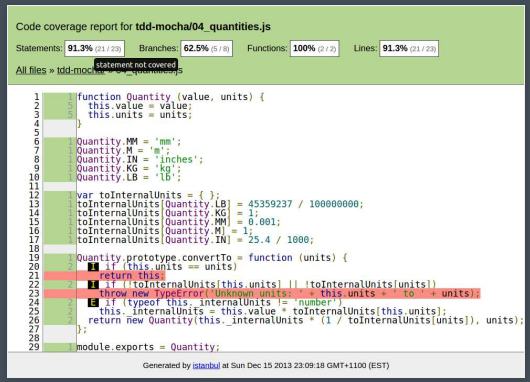
```
  Quantity
    ✓ should hold value and units
    conversions
      ✓ should convert from kilograms to pounds
      ✓ should convert from pounds to kilograms


  3 passing (4ms)

================================================================
Writing coverage object [/tmp/tdd-mocha/coverage/coverage.json]
Writing coverage reports at [/tmp/tdd-mocha/coverage]
================================================================

============================== Coverage summary ================
Statements   : 91.3% ( 21/23 )
Branches     : 62.5% ( 5/8 )
Functions    : 100% ( 2/2 )
Lines        : 91.3% ( 21/23 )
================================================================
```

CODE COVERAGE WITH ISTANBUL



Code coverage report for **tdd-mocha/04_quantities.js**

Statements: **91.3%** (21 / 23) Branches: **62.5%** (5 / 8) Functions: **100%** (2 / 2) Lines: **91.3%** (21 / 23)

All files » tdd-mocha/ 04_quantities.js

statement not covered

```
1   1 function Quantity (value, units) {
2   3   this.value = value;
3   3   this.units = units;
4     }
5
6   1 Quantity.MM = 'mm';
7   1 Quantity.M = 'm';
8   1 Quantity.IN = 'inches';
9   1 Quantity.KG = 'kg';
10  1 Quantity.LB = 'lb';
11
12  1 var toInternalUnits = { };
13  1 toInternalUnits[Quantity.LB] = 45359237 / 100000000;
14  1 toInternalUnits[Quantity.KG] = 1;
15  1 toInternalUnits[Quantity.MM] = 0.001;
16  1 toInternalUnits[Quantity.M] = 1;
17  1 toInternalUnits[Quantity.IN] = 25.4 / 1000;
18
19  1 Quantity.prototype.convertTo = function (units) {
20  2   E if (this.units == units)
21  0     return this;
22  2   E if (!toInternalUnits[this.units] || !toInternalUnits[units])
23  0     throw new TypeError('Unknown units: ' + this.units + ' to ' + units);
24  2   E if (typeof this._internalUnits != 'number')
25  2     this._internalUnits = this.value * toInternalUnits[this.units];
26  2   return new Quantity(this._internalUnits * (1 / toInternalUnits[units]), units);
27    };
28
29  1 module.exports = Quantity;
```

Generated by istanbul at Sun Dec 15 2013 23:09:18 GMT+1100 (EST)

Created ./coverage/lcov-report/index.html for each test run

- What do we need to do to improve coverage?
- Where is Istanbul inadequate?