2025/12/13 20:37 1/12 Mocha

Mocha

```
LearnBoost TJ Holowaychuk가
expresso
Mocha TDD BDD
                                       API

    CI

                    exit
   • tty가
                                  uncaught exception
   growl
   • mocha.opts
   • node
   done()
                                  가
           assertion
         가 9
         가
                      DSL
   • before, after, before each, after each
                       (tranpiler)
                                  (coffeed-script6 )

    TextMate
```

\$ sudo npm install -g mocha

Assertion

가 assertion 가 assertion .

- **should.js**: BDD style shown throughout these docs
- expect.js : expect() style assertions
- **chai**: expect(), assert() and should style assertions
- **better-assert** : c-style self-documenting assert()

BDD가

BDD

```
describe('BDD style', function() {
  before(function() {
   // excuted before test suite
  });
  after(function() {
   // excuted after test suite
  });
  beforeEach(function() {
   // excuted before every test
  });
  afterEach(function() {
    // excuted after every test
  });
  describe('#example', function() {
    it('this is a test.', function() {
     // write test logic
   });
 });
});
```

TDD

```
TDD suite() test() / suiteSetup(), suiteTeardown(), setup(), teardown() .
```

2025/12/13 20:37 3/12 Mocha

```
suite('TDD Style', function() {
  suiteSetup(function() {
    // excuted before test suite
  });
  suiteTeardown(function() {
   // excuted after test suite
  });
  setup(function() {
   // excuted before every test
  });
  teardown(function() {
   // excuted before every test
  });
  suite('#example', function() {
    test('this is a test', function() {
     // write test logic
   });
 });
});
```

exports, QUnit

Synchronous code

```
it() assertion . node.js assert
```

```
var assert = require('assert');

describe('Example', function() {
    describe('calculation', function() {
        it('1+1 should be 2', function() {
            assert.equal(1+1, 2);
        });
    });
});
```

Asynchronous code

```
it()
                           done
                                                 가 . assertion
                          done()
                                                                 done()
               . done()
                                                  2000ms
 . (
                가 )
var assert = require('assert')
    fs = require('fs');
describe('Example', function() {
  describe('calculation', function() {
    it('1+1 should be 2', function(done) {
       fs.readFile('example.txt', function(err, data) {
        done();
      });
    });
  });
});
done()
          node.js
var assert = require('assert')
    fs = require('fs');
describe('Example', function() {
  describe('calculation', function() {
    it('1+1 should be 2', function(done) {
      fs.readFile('example.txt', done);
    });
  });
});
         test
                                   test
$ mocha
                                              가
$ mocha test.js
```

2025/12/13 20:37 5/12 Mocha

mocha

-h	mocha 가 .		
-w, -watch	가 .		
-compilers <ext>:<module></module></ext>	transpilercompilers coffee:coffee-script .		
-b, -bail			
-d, -debug	node.js .		
-globas <names></names>	. mocha		
-ignore-leaks	가		
-r, -require <name></name>	require() Object.prototype . assertion should.js var should = require('should'); 가 -require should .(should가 가 .) module.exports require() .		
-u, -ui <name></name>	. bdd . tdd, exports, qunit		
-t, -timeout <ms></ms>	-timeout 2s -timeout 2000 . 2 .		
-s, -slow <ms></ms>	75ms .		
-g, -grep <pattern></pattern>			
-G, -growl	growl .		
-interfaces			
-reporters	·		

가 -R, -reporter <name></name>		
dot .		. (
mocha)	

- dot dot matrix
- doc html documentation
- spec hierarchical spec list
- json single json object
- progress progress bar
- list spec-style listing
- tap test-anything-protocol
- landing unicode landing strip
- xunit xunit reportert
- teamcity teamcity ci support
- html-cov HTML test coverage
- json-cov JSON test coverage
- min minimal reporter (great with -watch)
- json-stream newline delimited json events

mocha

가

• markdown - markdown documentation (github flavour)

가

mocha.opts

```
./test/mocha.opts
--require should
--reporter dot
--ui bdd
        mocha.opts
                                  mocha
                                                   mocha -require should -reporter dot -ui
bdd
                           가
$ mocha --reporter list --growl
                                        가
                                              mocha.opts
mocha.opts
      Mocha + Chai(Assertion
                                  $ npm install chai
testPrj
                                                                chai
                                                                                   lib
                      . lib
                                                     가
    test
                                                         test
lib
tags.js
```

```
exports = module.exports = {};
exports.parse = function(args, defaults, replacements) {
    var options = {};
    if (typeof defaults === "object" && !(defaults instanceof Array)) {
        options = defaults
    if (typeof replacements === "object" && !(defaults instanceof
Array)) {
        for (var i in args) {
            var arg = args[i];
```

2025/12/13 20:37 7/12 Mocha

```
if (arg.charAt(0) === "-" \&\& arg.charAt(1) != "-") {
            arg = arg.substr(1);
            if (arg.index0f("=") !== -1) {
                arg = arg.split("=");
                var keys = arg.shift();
                var value = arg.join("=");
                arg = keys.split("");
                var key = arg.pop();
                if (replacements.hasOwnProperty(key)) {
                    key = replacements[key];
                }
                args.push("--" + key + "=" + value);
            } else {
                arg = arg.split("");
            arg.forEach(function(key){
                if (replacements.hasOwnProperty(key)) {
                    key = replacements[key];
                args.push("--" + key);
            });
        }
   }
}
for (var i in args) { //Cycle through args
    var arg = args[i];
    //Check if Long formed tag
    if (arg.substr(0, 2) === "--") {
        arg = arg.substr(2);
        //Check for equals sign
        if (arg.index0f("=") !== -1) {
            arg = arg.split("=");
            var key = arg.shift();
            var value = arg.join("=");
            if (/^[0-9]+$/.test(value)) {
                value = parseInt(value, 10);
            options[key] = value;
        } else {
            options[arg] = true;
return options;
```

search.js

```
var fs = require("fs");
exports = module.exports = {};
//A Modified Snippet from Christopher Jeffrey
http://stackoverflow.com/questions/5827612/node-js-fs-readdir-recursive
-directory-search
exports.scan = function(dir, depth, done) {
    depth--;
    var results = [];
    fs.readdir(dir, function(err, list) {
        if (err) return done(err);
        var i = 0;
        (function next() {
            var file = list[i++];
            if (!file) return done(null, results);
            file = dir + '/' + file;
            fs.stat(file, function(err, stat) {
                if (stat && stat.isDirectory()) {
                    if (depth !== 0) {
                        var ndepth = (depth > 1) ? depth-1 : 1;
                        exports.scan(file, ndepth, function(err, res) {
                             results = results.concat(res);
                            next():
                        });
                    } else {
                        next();
                    }
                } else {
                    results.push(file);
                    next();
            });
       })();
   });
};
exports.match = function(query, files){
    var matches = [];
    files.forEach(function(name) {
        if (name.index0f(query) !== -1) {
            matches.push(name);
        }
    });
    return matches;
```

2025/12/13 20:37 9/12 Mocha

test

tagsSpec.js

```
var expect = require("chai").expect;
var tags = require("../lib/tags.js");
describe("Tags", function(){
    describe("#parse()", function(){
        it("should parse long formed tags and convert numbers",
function(){
            var args = ["--depth=4", "--hello=world"];
            var results = tags.parse(args);
            expect(results).to.have.a.property("depth", 4);
            expect(results).to.have.a.property("hello", "world");
        });
        it("should fallback to defaults", function(){
            var args = ["--depth=4", "--hello=world"];
            var defaults = { depth: 2, foo: "bar" };
            var results = tags.parse(args, defaults);
            var expected = {
                depth: 4,
                foo: "bar",
                hello: "world"
            };
            expect(results).to.deep.equal(expected);
        });
        it("should accept tags without values as a bool", function(){
            var args = ["--searchContents"];
            var results = tags.parse(args);
            expect(results).to.have.a.property("searchContents", true);
        });
        it("should accept short formed tags", function(){
            var args = ["-sd=4", "-h"];
            var replacements = {
                s: "searchContents",
                d: "depth",
                h: "hello"
            };
            var results = tags.parse(args, {}, replacements);
            var expected = {
                searchContents: true,
```

```
depth: 4,
    hello: true
};

expect(results).to.deep.equal(expected);
});
});
```

```
Synchronous code
                                        . it
                                                                                        Tags
                              parse()
                                                                    describe
                      ...)
  it
                               가
                                                                                     가
                     parse
                                                           parsing
                                                                   argument
  chai
              expect
                                                     parse
                                                                            가
                (parseInt
                                         )
                                                     가
```

searchSpec.js

```
var expect = require("chai").expect;
var search = require("../lib/search.js");
var fs = require("fs");
describe("Search", function(){
    describe("#scan()", function(){
        before(function() {
            if (!fs.existsSync(".test files")) {
                fs.mkdirSync(".test_files");
                fs.writeFileSync(".test_files/a", "");
                fs.writeFileSync(".test files/b", "");
                fs.mkdirSync(".test_files/dir");
                fs.writeFileSync(".test files/dir/c", "");
                fs.mkdirSync(".test files/dir2");
                fs.writeFileSync(".test_files/dir2/d", "");
            }
        });
        after(function() {
            fs.unlinkSync(".test_files/dir/c");
            fs.rmdirSync(".test_files/dir");
            fs.unlinkSync(".test files/dir2/d");
            fs.rmdirSync(".test_files/dir2");
            fs.unlinkSync(".test files/a");
            fs.unlinkSync(".test_files/b");
            fs.rmdirSync(".test files");
        });
        it("should retrieve the files from a directory", function(done)
            search.scan(".test_files", 0, function(err, flist){
                expect(flist).to.deep.equal([
```

2025/12/13 20:37 11/12 Mocha

```
".test_files/a",
                    ".test_files/b",
                    ".test files/dir/c",
                    ".test files/dir2/d"
                ]);
                done();
            });
        });
        it("should stop at a specified depth", function(done) {
            search.scan(".test_files", 1, function(err, flist) {
                expect(flist).to.deep.equal([
                    ".test_files/a",
                    ".test files/b",
                ]);
                done();
            });
        });
    });
    describe("#match()", function(){
        it("should find and return matches based on a query",
function(){
            var files = ["hello.txt", "world.js", "another.js"];
            var results = search.match(".js", files);
            expect(results).to.deep.equal(["world.js", "another.js"]);
            results = search.match("hello", files);
            expect(results).to.deep.equal(["hello.txt"]);
        });
   });
});
```

```
Asynchronous code
                                         . scan()
    before()
                   after()
                                                    scan()
fs
                                       가
                                                                               .test files
            before()
                                                                           . scan()
                                                   가
                                                                        가
                   done
                                                             done()
                          before(), after()
                                                       synchronous code
                         '-' ' Testing in Node.js
                                           input, output
```

Webstorm

Debugging mocha unit tests with WebStorm step by step

From:

http://obg.co.kr/doku/ - OBG WiKi

Permanent link:

http://obg.co.kr/doku/doku.php?id=programming:javascript:tdd:mocha

Last update: 2020/11/29 14:09

