

Chai.js should

```
object.should.equal(expected)
object.should.eql(expected)
object.should.deep.equal(expected) // same as .eql
object.should.be.a('string')
object.should.include(val)
object.should.be.ok(val)
object.should.be.true
object.should.be.false
object.should.be.null
object.should.be.undefined
object.should.be.empty
object.should.be.arguments
object.should.be.function
object.should.be.instanceOf
object.should.gt(5) # or .above .greaterThan
object.should.gte # or .at.least
object.should.lt(5) # or .below
object.should.respondTo('bar')
object.should.satisfy (n) -> n > 0
object.should.have.members([2, 3, 4])
object.should.have.keys(['foo'])
object.should.have.key('foo')
object.should.exist
```

```
require('chai').should();
//actually call the function, add "should" method to prototype of object
```

Chai.js expect

```
expect(object).equal(expected)
expect(object).eql(expected)
expect(object).deep.equal(expected) // same as .eql
expect(object).be.a('string')
expect(object).include(val)
expect(object).be.ok(val)
expect(object).be.true
```

Chai.js expect (cont)

```
expect(object).be.false
expect(object).be.null
expect(object).be.undefined
expect(object).be.empty
expect(object).be.arguments
expect(object).be.function
expect(object).be.instanceOf
expect(object).gt(5) # or .above .greaterThan
expect(object).gte # or .at.least
expect(object).lt(5) # or .below
expect(object).respondTo('bar')
expect(object).satisfy (n) -> n > 0
expect(object).have.members([2, 3, 4])
expect(object).have.keys(['foo'])
expect(object).have.key('foo')
expect(object).exist
expect(object).(-> ...).throw /not a function/
```

Chai.js asserts

```
assert(val)
assert.fail(actual, expected)
assert.ok(val) // is truthy
assert.equal(actual, expected) // 'compare with =='
assert.strictEqual
assert.deepEqual
assert.isTrue
assert.isFalse
assert.isNull
assert.isNotNull
assert.isUndefined
assert.isDefined
assert.isFunction
assert.isObject
```

Chai.js asserts (cont)

```
assert.isArray
assert.isString
assert.isNumber
assert.isBoolean
assert.typeOf(/tea/, 'regexp') // Object.prototype.toString()
assert.instanceOf(chai, Tea)
assert.include([ a,b,c ], a)
assert.match(val, /regexp/)
assert.property(obj, 'tea') // 'tea' in object
assert.deepProperty(obj, 'tea.green')
assert.propertyVal(person, 'name', 'John')
assert.deepPropertyVal(post, 'author.name', 'John')
assert.lengthOf(object, 3)
assert.throws(function() { ... })
assert.doesNotThrow
assert.operator(1, '<', 2)
assert.closeTo(actual, expected)
```

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

Sinon-chai

```
expect(spy).called
expect(spy).calledOnce
expect(spy).calledTwice
expect(spy).calledThrice
expect(spy).calledBefore(spy2)
expect(spy).calledAfter(spy2)
expect(spy).calledWithNew
expect(spy).alwaysCalledWithNew
expect(spy).calledOn(context)
expect(spy).alwaysCalledOn(context)
expect(spy).calledWith(...args)
expect(spy).alwaysCalledWith(...args)
expect(spy).calledWithExactly(...args)
```



By **Andrey (apk)**
cheatography.com/apk/
kucherenko.org/

Published 21st November, 2015.
 Last updated 17th December, 2015.
 Page 1 of 3.

Sponsored by **Readability-Score.com**
 Measure your website readability!
<https://readability-score.com>

Sinon-chai (cont)

`expect(spy).alwaysCalledWithExactly(...args)`

`expect(spy).calledWithMatch(...args)`

`expect(spy).alwaysCalledWithMatch(...args)`

`expect(spy).returned(val)`

`expect(spy).alwaysReturned(val)`

`expect(spy).threw(errorObjOrErrorTypeStringOrNothing)`

`expect(spy).alwaysThrew(errorObjOrErrorTypeStringOrNothing)`

`spy.should.have.been.called`

`spy.should.have.been.calledOnce`

`spy.should.have.been.calledTwice`

`spy.should.have.been.calledThrice`

`spy1.should.have.been.calledBefore(spy2)`

`spy1.should.have.been.calledAfter(spy2)`

`spy.should.have.been.calledWithNew`

`spy.should.always.have.been.calledWithNew`

`spy.should.have.been.calledOn(context)`

`spy.should.always.have.been.calledOn(context)`

`spy.should.have.been.calledWith(...args)`

`spy.should.always.have.been.calledWith(...args)`

`spy.should.always.have.been.calledWithExactly(...args)`

`spy.should.always.have.been.calledWithExactly(...args)`

`spy.should.have.been.calledWithMatch(...args)`

`spy.should.always.have.been.calledWithMatch(...args)`

`spy.should.have.returned(returnVal)`

`spy.should.have.always.returned(returnVal)`

`spy.should.have.threw(errorObjOrErrorTypeStringOrNothing)`

Sinon-chai (cont)

`spy.should.have.always.thrown(errorObjOrErrorTypeStringOrNothing)`

```
var sinon = require('sinon');
require('chai').use(require('sinon-chai'));
```

Note that you can negate any assertion with Chai's `.not`. E. g. for `notCalled` use `spy.should.have.not.been.called`.

Mocha BDD

```
mocha.setup('bdd');
describe.only('something',
function() {
  beforeEach(function() {
  });
  it.skip('should work',
function() {
  });
  it('should save',
function(done) {
    var user = new User();
    user.save(function(err) {
      if (err) throw err;
      done();
    });
  });
});
```

Mocha TDD

```
mocha.setup('tdd');
suite('something', function() {
  setup(function() {
  });
  test('should work', function() {
  });
  teardown(function() {
  });
});
```

Sinon.js Spy/stub properties

```
spy
  .args //=> [ [...], [...] ] one
per call
  .thisValues
  .returnValues
  .called //=> true
  .notCalled
  .callCount
  .calledOnce
  .calledTwice
  .calledThrice
  .getCalls() //=> Array
  .getCall(0)
  .firstCall
```

Sinon.JS Sandbox

```
beforeEach(function() {
  global.env =
require('sinon').sandbox.create();
});
afterEach(function() {
  global.env.restore();
});
```

Sinon.js Fake Server, XHR and date

```
$.get('/file.json', ...);
server.requests[0].respond(
  200,
  { "Content-Type":
"application/json" },
  JSON.stringify({ id: 1, text:
"Provide examples", done: true })
);
server.restore();
xhr =
sinon.useFakeXMLHttpRequest();
xhr.restore();
sinon.useFakeTimers(+new
Date(2011,9,1));
```



By **Andrey (apk)**
cheatography.com/apk/
kucherenko.org/

Published 21st November, 2015.
 Last updated 17th December, 2015.
 Page 2 of 3.

Sponsored by **Readability-Score.com**
 Measure your website readability!
<https://readability-score.com>

Sinon.js spies

```
fn = sinon.spy();  
fn();  
fn.calledOnce == true  
fn.callCount == 1
```

Sinon.js Spying/stubbing

```
sinon.spy($, 'ajax')  
$.ajax();  
$.ajax.calledOnce == true  
sinon.stub($, 'ajax', function () {  
  ... }); // function optional  
$.ajax.calledWithMatch({ url: '/x'  
});  
$.ajax.restore();
```

Sinon.js mocks expectations

```
var mock = sinon.mock(obj);  
  
var expectation = mock.expects("method");  
expectation.atLeast(number);  
expectation.atMost(number);  
expectation.never();  
expectation.once();  
expectation.twice();  
expectation.thrice();  
expectation.exactly(number);  
expectation.withArgs(arg1, arg2, ...);  
expectation.withExactArgs(arg1, arg2, ...);  
expectation.on(obj);  
expectation.verify();  
mock.restore();  
mock.verify();  
  
sinon.mock(jQuery).expects("ajax").  
atLeast(2).atMost(5);  
jQuery.ajax.verify();
```

Sinon.js stubs

```
stub = sinon.stub().returns(42);  
stub() == 42  
stub  
  .withArgs(42).returns(1);  
  .withArgs(43).throws("TypeError")  
;  
stub  
  .returns(1);  
  .throws("TypeError");  
  .returnsArg(0); // Return 1st  
argument  
  .callsArg(0);
```



By **Andrey (apk)**
cheatography.com/apk/
kucherenko.org/

Published 21st November, 2015.
Last updated 17th December, 2015.
Page 3 of 3.

Sponsored by **Readability-Score.com**
Measure your website readability!
<https://readability-score.com>