*Chai Assertion Library*



const { assert } = require('chai')

assert(val)

assert.fail(actual, expected)

assert.ok(val) // is truthy

assert.equal(actual, expected)

assert.strictEqual(actual, expected)

assert.deepEqual(actual, expected)

assert.isTrue(val)

assert.isFalse(val)

assert.isNull(val)

assert.isNotNull(val)

assert.isUndefined(val)

assert.isDefined(val)

assert.isFunction(val)

assert.isObject(val)

assert.isArray(val)

assert.isString(val)

assert.isNumber(val)

assert.isBoolean(val)

assert.typeOf(/tea/, 'regexp')

assert.instanceOf(chai, Tea)

assert.include([ a,b,c ], a)

assert.match(val, /regexp/)

assert.property(obj, 'tea')

assert.deepProperty(obj, 'tea.green')

assert.propertyVal(person, 'name', 'John')

assert.deepPropertyVal(post, 'author.name', 'John')

assert.lengthOf(object, 3)

assert.throws(function() { ... })

assert.throws(function() { ... }, /reference error/)

assert.doesNotThrow

assert.operator(1, '<', 2)

assert.closeTo(actual, expected)

const { expect } = require('chai')

expect(object)

.to.equal(expected)

.to.eql(expected)

.to.deep.equal(expected)

.to.be.a('string')

.to.include(val)

.be.ok(val)

.be.true

.be.false

.to.exist

.to.be.null

.to.be.undefined

.to.be.empty

.to.be.arguments

.to.be.function

.to.be.instanceOf

.to.be.gt(5)

.to.be.gte(5)

.to.be.lt(5)

.to.respondTo('bar')

.to.satisfy((n) => n > 0)

.to.have.members([2, 3, 4])

.to.have.keys(['foo'])

.to.have.key('foo')

expect(() => { ··· })

.to.throw(/not a function/)

### NOT

expect(object).not.equal('x')

Should: chains

.to .be .been

.is .that .and

.have .with

.at .of .same

*Chai with jQuery*

global.jQuery = ···

chai.use(require('chai-jquery'))

expect($body)

.have.attr('foo')

.have.prop('disabled')

.have.css('background')

.have.css('background-color', '#ffffff')

.have.data('foo')

.have.class('active')

.have.id('id')

.have.html('<em>hi</em>')

.have.text('hello')

.have.value('2013')

.be.visible

.be.hidden

.be.checked

.be.selected

.be.enabled

.be.disabled

.be.empty

.to.exist

.to.contain('text')

.to.have('.selector')