**TDD Assertions**

These assertions are available for TDD assertions (assert). You can see the entire list of available Chai assertions [here](http://chaijs.com/api/assert/).

| **Assertion** | **Example** |
| --- | --- |
| .isOk(*object*, *[message]*) | assert.isOk('everything', 'everything is ok') |
| .isNotOk(*object*, *[message]*) | assert.isNotOk(false, 'this will pass') |
| .equal(*actual*, *expected*, *[message]*) | assert.equal(3, 3, 'vals equal') |
| .notEqual(*actual*, *expected*, *[message]*) | assert.notEqual(3, 4, 'vals not equal') |
| .strictEqual(*actual*, *expected*, *[message]*) | assert.strictEqual(true, true, 'bools strict eq') |
| .notStrictEqual(*actual*, *expected*, *[message]*) | assert.notStrictEqual(5, '5', 'not strict eq') |
| .deepEqual(*actual*, *expected*, *[message]*) | assert.deepEqual({ id: '1' }, { id: '1' }) |
| .notDeepEqual(*actual*, *expected*, *[message]*) | assert.notDeepEqual({ id: '1' }, { id: '2' }) |
| .isAbove(*valueToCheck*, *valueToBeAbove*, *[message]*) | assert.isAbove(6, 1, '6 greater than 1') |
| .isAtLeast(*valueToCheck*, *valueToBeAtLeast*, *[message]*) | assert.isAtLeast(5, 2, '5 gt or eq to 2') |
| .isBelow(*valueToCheck*, *valueToBeBelow*, *[message]*) | assert.isBelow(3, 6, '3 strict lt 6') |
| .isAtMost(*valueToCheck*, *valueToBeAtMost*, *[message]*) | assert.isAtMost(4, 4, '4 lt or eq to 4') |
| .isTrue(*value*, *[message]*) | assert.isTrue(true, 'this val is true') |
| .isNotTrue(*value*, *[message]*) | assert.isNotTrue('tests are no fun', 'val not true') |
| .isFalse(*value*, *[message]*) | assert.isFalse(false, 'val is false') |
| .isNotFalse(*value*, *[message]*) | assert.isNotFalse('tests are fun', 'val not false') |
| .isNull(*value*, *[message]*) | assert.isNull(err, 'there was no error') |
| .isNotNull(*value*, *[message]*) | assert.isNotNull('hello', 'is not null') |
| .isNaN(*value*, *[message]*) | assert.isNaN(NaN, 'NaN is NaN') |
| .isNotNaN(*value*, *[message]*) | assert.isNotNaN(5, '5 is not NaN') |
| .exists(*value*, *[message]*) | assert.exists(5, '5 is not null or undefined') |
| .notExists(*value*, *[message]*) | assert.notExists(null, 'val is null or undefined') |
| .isUndefined(*value*, *[message]*) | assert.isUndefined(undefined, 'val is undefined') |
| .isDefined(*value*, *[message]*) | assert.isDefined('hello', 'val has been defined') |
| .isFunction(*value*, *[message]*) | assert.isFunction(x => x \* x, 'val is func') |
| .isNotFunction(*value*, *[message]*) | assert.isNotFunction(5, 'val not funct') |
| .isObject(*value*, *[message]*) | assert.isObject({num: 5}, 'val is object') |
| .isNotObject(*value*, *[message]*) | assert.isNotObject(3, 'val not object') |
| .isArray(*value*, *[message]*) | assert.isArray(['unit', 'e2e'], 'val is array') |
| .isNotArray(*value*, *[message]*) | assert.isNotArray('e2e', 'val not array') |
| .isString(*value*, *[message]*) | assert.isString('e2e', 'val is string') |
| .isNotString(*value*, *[message]*) | assert.isNotString(2, 'val not string') |
| .isNumber(*value*, *[message]*) | assert.isNumber(2, 'val is number') |
| .isNotNumber(*value*, *[message]*) | assert.isNotNumber('e2e', 'val not number') |
| .isFinite(*value*, *[message]*) | assert.isFinite('e2e', 'val is finite') |
| .isBoolean(*value*, *[message]*) | assert.isBoolean(true, 'val is bool') |
| .isNotBoolean(*value*, *[message]*) | assert.isNotBoolean('true', 'val not bool') |
| .typeOf(*value*, *name*, *[message]*) | assert.typeOf('e2e', 'string', 'val is string') |
| .notTypeOf(*value*, *name*, *[message]*) | assert.notTypeOf('e2e', 'number', 'val not number') |