### BDD Assertions

These chainers are available for BDD assertions (expect/should). Aliases listed can be used interchangeably with their original chainer. You can see the entire list of available BDD Chai assertions [here](http://chaijs.com/api/bdd/).

| **Chainer** | **Example** |
| --- | --- |
| not | expect(name).to.not.equal('Jane') |
| deep | expect(obj).to.deep.equal({ name: 'Jane' }) |
| nested | expect({a: {b: ['x', 'y']}}).to.have.nested.property('a.b[1]') expect({a: {b: ['x', 'y']}}).to.nested.include({'a.b[1]': 'y'}) |
| ordered | expect([1, 2]).to.have.ordered.members([1, 2]).but.not.have.ordered.members([2, 1]) |
| any | expect(arr).to.have.any.keys('age') |
| all | expect(arr).to.have.all.keys('name', 'age') |
| a(*type*)  **Aliases:** an | expect('test').to.be.a('string') |
| include(*value*)  **Aliases:** contain, includes, contains | expect([1,2,3]).to.include(2) |
| ok | expect(undefined).to.not.be.ok |
| true | expect(true).to.be.true |
| false | expect(false).to.be.false |
| null | expect(null).to.be.null |
| undefined | expect(undefined).to.be.undefined |
| exist | expect(myVar).to.exist |
| empty | expect([]).to.be.empty |
| arguments  **Aliases:** Arguments | expect(arguments).to.be.arguments |
| equal(*value*)  **Aliases:** equals, eq | expect(42).to.equal(42) |
| deep.equal(*value*) | expect({ name: 'Jane' }).to.deep.equal({ name: 'Jane' }) |
| eql(*value*)  **Aliases:** eqls | expect({ name: 'Jane' }).to.eql({ name: 'Jane' }) |
| greaterThan(*value*)  **Aliases:** gt, above | expect(10).to.be.greaterThan(5) |
| least(*value*) **Aliases:** gte | expect(10).to.be.at.least(10) |
| lessThan(*value*)  **Aliases:** lt, below | expect(5).to.be.lessThan(10) |
| most(*value*)  **Aliases:** lte | expect('test').to.have.length.of.at.most(4) |
| within(*start*, *finish*) | expect(7).to.be.within(5,10) |
| instanceOf(*constructor*)  **Aliases:** instanceof | expect([1, 2, 3]).to.be.instanceOf(Array) |
| property(*name*, *[value]*) | expect(obj).to.have.property('name') |
| deep.property(*name*, *[value]*) | expect(deepObj).to.have.deep.property('tests[1]', 'e2e') |
| ownProperty(*name*)  **Aliases:** haveOwnProperty, own.property | expect('test').to.have.ownProperty('length') |
| ownPropertyDescriptor(*name*)  **Aliases:** haveOwnPropertyDescriptor | expect({a: 1}).to.have.ownPropertyDescriptor('a') |
| lengthOf(*value*) | expect('test').to.have.lengthOf(3) |
| match(*RegExp*)  **Aliases:** matches | expect('testing').to.match(/^test/) |
| string(*string*) | expect('testing').to.have.string('test') |
| keys(*key1*, *[key2]*, *[...]*)  **Aliases:** key | expect({ pass: 1, fail: 2 }).to.have.keys('pass', 'fail') |
| throw(*constructor*)  **Aliases:** throws, Throw | expect(fn).to.throw(Error) |
| respondTo(*method*)  **Aliases:** respondsTo | expect(obj).to.respondTo('getName') |
| itself | expect(Foo).itself.to.respondTo('bar') |
| satisfy(*method*)  **Aliases:** satisfies | expect(1).to.satisfy((num) => { return num > 0 }) |
| closeTo(*expected*, *delta*)  **Aliases:** approximately | expect(1.5).to.be.closeTo(1, 0.5) |
| members(*set*) | expect([1, 2, 3]).to.include.members([3, 2]) |
| oneOf(*values*) | expect(2).to.be.oneOf([1,2,3]) |
| change(*function*)  **Aliases:** changes | expect(fn).to.change(obj, 'val') |
| increase(*function*)  **Aliases:** increases | expect(fn).to.increase(obj, 'val') |
| decrease(*function*)  **Aliases:** decreases | expect(fn).to.decrease(obj, 'val') |