Global Environment

`afterEach(fn)`	Called once after each test
`beforeEach(fn)`	Called once before each test
`afterAll(fn)`	Called only once after all the tests in describe finish
`beforeAll(fn)`	Called only once before all the tests in describe
`describe(name, fn)`	Test suite describer
`expect(value)`	Starts an expectation chain
`expect.extend(matchers)`	Adds more matchers to expect
`expect.<asymmetric-match>()`	Used as value to match to
`it(name, fn)`	Test describer (another alias for it is `test`)
`it.only(name, fn)`	Skip all tests after this one
`it.skip(name, fn)`	Skip this test
`fit(name, fn)`	Runs only this test
`jest`	The `jest` object
`require.requireActual(moduleName)`	Returns the actual module instead of a mock
`require.requireMock(moduleName)`	Returns a mock module instead of the actual module
`xdescribe/xit/xtest`	Makes the test pending
`fdescribe/fit`	Makes the test/suite the only one running

Writing assertions with expect

`.not`	Expects the opposite of the rest of the expectation.
`.toBe(value)`	Strict equality (===)
`.toEqual(value)`	Recursively checks the equality of all fields
`.toMatch(regexp)`	
`.toMatchObject(object)`	Matches a subset of the properties of an object

Writing assertions with expect (cont)

`.toMatchSnapshot()`	Matches the most recent snapshot (or creates one)
`.toBeDefined()`	
`.toBeUndefined()`	
`.toBeNull()`	
`.toBeTruthy()`	
`.toBeFalsy()`	
`.toBeGreaterThan(number)`	
`.toBeGreaterThanOrEqual(number)`	
`.toBeLessThan(number)`	
`.toBeLessThanOrEqual(number)`	
`.toBeCloseTo(number, numDigits)`	
`.toBeNaN()`	
`.toBeInstanceOf(Class)`	
`.toContain(item)`	Strict equality (===)
`.toContainEqual(item)`	Recursively checks the equality of all fields
`.toHaveLength(number)`	
`.toThrow()`	
`.toThrowError(error)`	`error` can be a string, RegExp or Class
`.toThrowErrorMatchingSnapshot()`	Will expect an error while matching (the snapshot will be the error's message)
`.toHaveBeenCalled()`	
`.toHaveBeenCalledTimes(number)`	
`.toHaveBeenCalledWith(arg1, arg2, ...)`	
`.toHaveBeenLastCalledWith(arg1, arg2, ...)`	

Mock functions

`jest.fn()`	Creates a mock function
`mockFn.mock.calls`	An array that represents all calls that have been made into this mock function
`mockFn.mock.instances`	An array that contains all the object instances
`mockFn.mockClear()`	Resets all information stored in the above arrays.
`mockFn.mockReset()`	Resets all information stored in the mock
`mockFn.mockImplementation(fn)`	Sets a function that should be used as the implementation of the mock
`mockFn.mockImplementationOnce(fn)`	Sets a function that should be used as the implementation of the mock for one call
`mockFn.mockReturnThis()`	Suger for using `jest.fn(()=>this)`
`mockFn.mockReturnValueOnce(value)`	

The jest object

`jest.resetAllMocks()`	Resets the state of all mocks
`jest.clearAllTimers()`	Removes any pending timers from the timer system
`jest.disableAutomock()`	Disables automatic mocking in the module loader
`jest.enableAutomock()`	Enables automatic mocking in the module loader
`jest.fn(?implementation)`	Returns a new, unused mock function
`jest.isMockFunction(fn)`	Determines if the given function is a mocked function

The jest object (cont)

`jest.genMockFromModule(moduleName)`	Generate a mocked version of the module for you
`jest.mock(moduleName, ?factory, ?options)`	Mocks a module with an auto-mocked version
`jest.resetModules()`	Resets the module registry - the cache of all required modules
`jest.runAllTicks()`	Exhausts the **micro**-task queue
`jest.runAllTimers()`	Exhausts the **macro**-task queue
`jest.runAllImmediates()`	Exhausts all tasks queued by `setImmediate()`
`jest.runTimersToTime(msToRun)`	Executes only the macro task queue
`jest.runOnlyPendingTimers()`	Executes only the macro-tasks that are currently pending
`jest.setMock(moduleName, moduleExports)`	Explicitly supplies the mock object that the module system should return for the specified module
`jest.unmock(moduleName)`	Indicates that the module system should never return a mocked version of the specified module
`jest.useFakeTimers()`	Instructs Jest to use fake versions of the standard timer functions
`jest.useRealTimers()`	Instructs Jest to use the real versions of the standard timer functions

By **Eli Sherer** (elisherer)

cheatography.com/elisherer/

Not published yet.
Last updated 30th January, 2017.
Page 2 of 2.