Jasmine Cheatsheet

Revision : 1.0 Modified by: Myles Kadusale

Note:
label - a string that describes the code block
function - an anonymous function or function

Jasmine Statements

describe(label, function)
*a group of test(s) / **Test Suite**

it(label, function)
*test block / **Spec**

beforeEach(function)
*run prior each **it** statement block

afterEach(function)
*run after each **it** statement block

xdescribe(label, function)
*a Test **Suite** that will not be performed
*instead of commenting out all the code under a test suite just add x

xit(label, function)
*a Test **Spec** that will not be performed
*instead of commenting out all the code under a test spec just add x

How to create Custom Matcher

```
describe('Hello world', function() {

  beforeEach(function() {
    jasmine.addMatchers({
      toBeDivisibleByTwo: function() {
        return (this.actual % 2) === 0;
      }
    });
  });

  it('is divisible by 2', function() {
    expect(gimmeANumber()).toBeDivisibleByTwo();
  });

});
```

Sample Spec

```
it('should increment a variable', function () {
  var foo = 0;
  foo++;
  expect(foo).toEqual(1); // passes because foo == 1
});
```

Creating Spies

```
stub = jasmine.createSpy('stub');
stub("hello");
expect(whatAmI.identity).toEqual("stub");
expect(whatAmI).toHaveBeenCalled();
```

Event Spies

```
spyOnEvent($('#some_element'), 'click');
$('#some_element').click();
expect('click').toHaveBeenPreventedOn($('#some_element'));
expect('click').toHaveBeenTriggeredOn($('#some_element'));
```

Note:
use **.not**. to negate the condition like **expect**(x).**not**.**toEqual**(y)

Jasmine Matchers

expect(x).**toEqual**(y);
*compares objects or primitives and passes if they are equivalent

expect(x).**toBe**(y);
*compares objects or primitives with ===

expect(x).**toMatch**(y);
*compares x to a string or regex and passes if they match

expect(x).**toBeDefined**();
*passes if x is not undefined

expect(x).**toBeUndefined**();
*passes if x is undefined

expect(x).**toBeNull**();
*passes if x is null

expect(x).**toBeTruthy**();
*passes if x is true

expect(x).**toBeFalsy**();
*passes if x is false

expect(x).**toContain**(y);
*passes if (x) array or string contains (y)

expect(x).**toBeLessThan**(y);
*passes if x is less than y

expect(x).**toBeGreaterThan**(y);
*passes if x is greater than y

expect(fn()).**toThrow**(e);
*passes if function fn throws exception e when executed

expect(x).**toBeCloseTo**(y, precision);
*checks that x is equal to y up to a given level of decimal precision

Jasmine Spies Matchers

expect(fn).**toHaveBeenCalled**();
*passes if fn is a spy and was called

expect(fn).**toHaveBeenCalledWith**(arg1, arg2,..);
*to verify if fn was called with the specific arg(s)

spyOn(obj, method_string).**andReturn**(value)
*sets the return value when spy is called

spyOn(obj, method_string).**andCallThrough**()
*spies on all calls and calls the actual function spied on

spyOn(obj, method_string).**andThrow**(exception)
*throws passed exception when spy is called

spyOn(obj, method_string).**andCallFake**(function)
*calls passed function when spy is called

Note:
Spies are automatically removed after each spec. They may be set in the beforeEach function.

Jasmine Doubles/Spies

spyOn(obj, method_string) - assumes obj.method_string is a function
obj.stubbed.**calls** - returns array
obj.stubbed.**mostRecentCall** - call object
obj.stubbed.calls[0].**args** - return array

method.**argsForCall**[i] - returns array
*to get all arguments for all calls that have been made by a spy to i
method.**callCount**
*to know how many times the method was called
method.**mostRecentCall**.**args**
*to know what were the arguments to the last call
method.**reset**()
*to reset all the calls made to the spy so far

jasmine.**createSpyObj**(baseName, methods[])

```
tape = jasmine.createSpyObj('tape', ['play', 'pause', 'stop', 'rewind']);
tape.play();
tape.pause();
```

jasmine.**any**(constructor/class_name)

```
expect(12).toEqual(jasmine.any(Number));
```

jasmine.**createSpy**(id_string)
*create a bare spy that has no implementation behind it

```
var dummy = jasmine.createSpy('dummy');
$('#myButton').click(dummy);
```

Sample Test Suites

```
describe('Calculator', function () {
  var counter = 0;

  it('can add a number', function () {
    counter = counter + 2;  // counter was 0 before
    expect(counter).toEqual(2);
  });

  it('can multiply a number', function () {
    counter = counter * 5;  // counter was 2 before
    expect(counter).toEqual(10);
  });
```

Sample Test Suites with nested Describes

```
describe('some suite', function () {

  var suiteWideFoo;

  beforeEach(function () {
    suiteWideFoo = 0;
  });

  describe('some nested suite', function() {
    var nestedSuiteBar;
    beforeEach(function() {
      nestedSuiteBar=1;
    });

    it('nested expectation', function () {
      expect(suiteWideFoo).toEqual(0);
      expect(nestedSuiteBar).toEqual(1);
    });

  });

  it('top-level describe', function () {
    expect(suiteWideFoo).toEqual(0);
    expect(nestedSuiteBar).toEqual(undefined);
  });
});
```