|  |
| --- |
| //A spy will intercept a function:  //SRC: Testing your JavaScript with Jasmine spies , URL: https://www.youtube.com/watch?v=0EXBpsz9Bwc  //AHK: **[?spyon]** |
| var myObj = {  save: function() {  //...  },  getQuantity: function(){  return 5;  }  }  describe("Spies!!!", function(){   |  | | --- | | // @2:38 in video:  it("should spy on save() method", function(){  var spy = spyOn(myObj, 'save');  myObj.save();  expect(spy).toHaveBeenCalled();  }); | | // @3:36 in video:  //By default, spies don't relay/return the return value of the  //functions they intercept. So you'll have to mock out return value.  it("should spy on getQuantity", function(){  var spy = spyOn(myObj, "getQuantity").andReturn(10);  expect(myObj.getQuantity()).toEqual(10);  }); | | // @5:20: Not only can you intercept functions, you can also  // intercept + replace them with stub methods.  it("should spy on getQuantity fake", function(){  var spy = spyOn(myObj, "getQuantity").andCallFake(function(){  console.log("returning 20");  return 20;  });  expect(myObj.getQuantity()).toEqual(20);  }); | | // @7:12: If you just want to monitor if function was called, and how  // many times, but want it to actually return it's actual implementation  // value, use ".andCallThrough"  it("should spy on getQuantity callthru", function(){  var spy = spyOn(myObj, "getQuantity").andCallThrough();  expect(myObj.getQuantity()).toEqual(5);  expect(spy).toHaveBeenCalled();  }); | | //@8:56 : Throwing stuff!  if("Should spy on getQuantity throw", function(){  var spy = spyOn(myObj, "getQuantity").andThrow(new Error("problem"));  var qty;  try{  qty = myObj.getQuantity();  }catch{  qty = 100;  }  expect(qty).toEqual(100);  }); |   }); |