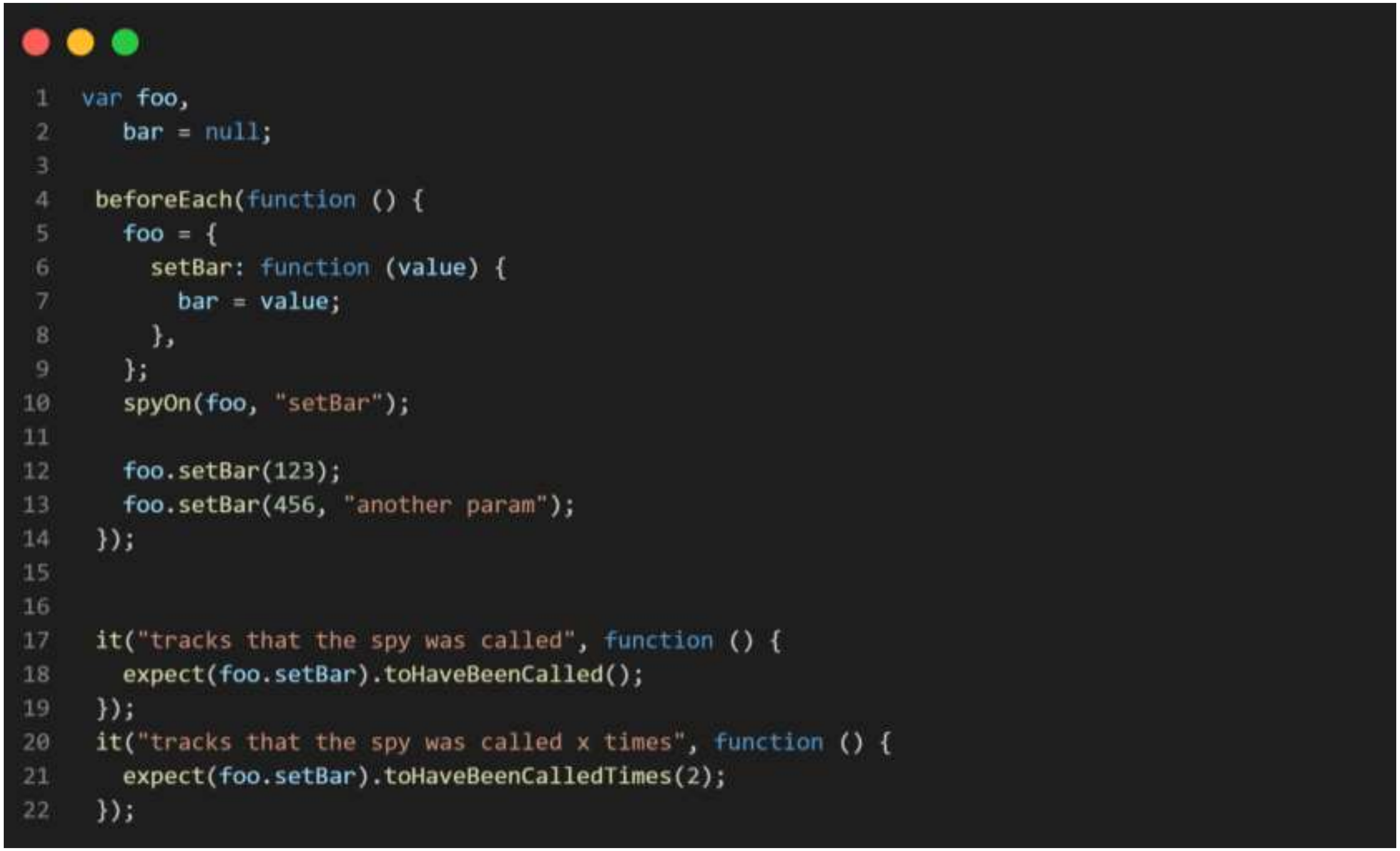
Jasmine

spy

Jasmine has test double functions called spies. A spy can stub any function and tracks calls to it and all arguments.

A spy only exists in the describe or it block in which it is defined, and will be removed after each spec. There are special matchers for interacting with spies

Spy Example :



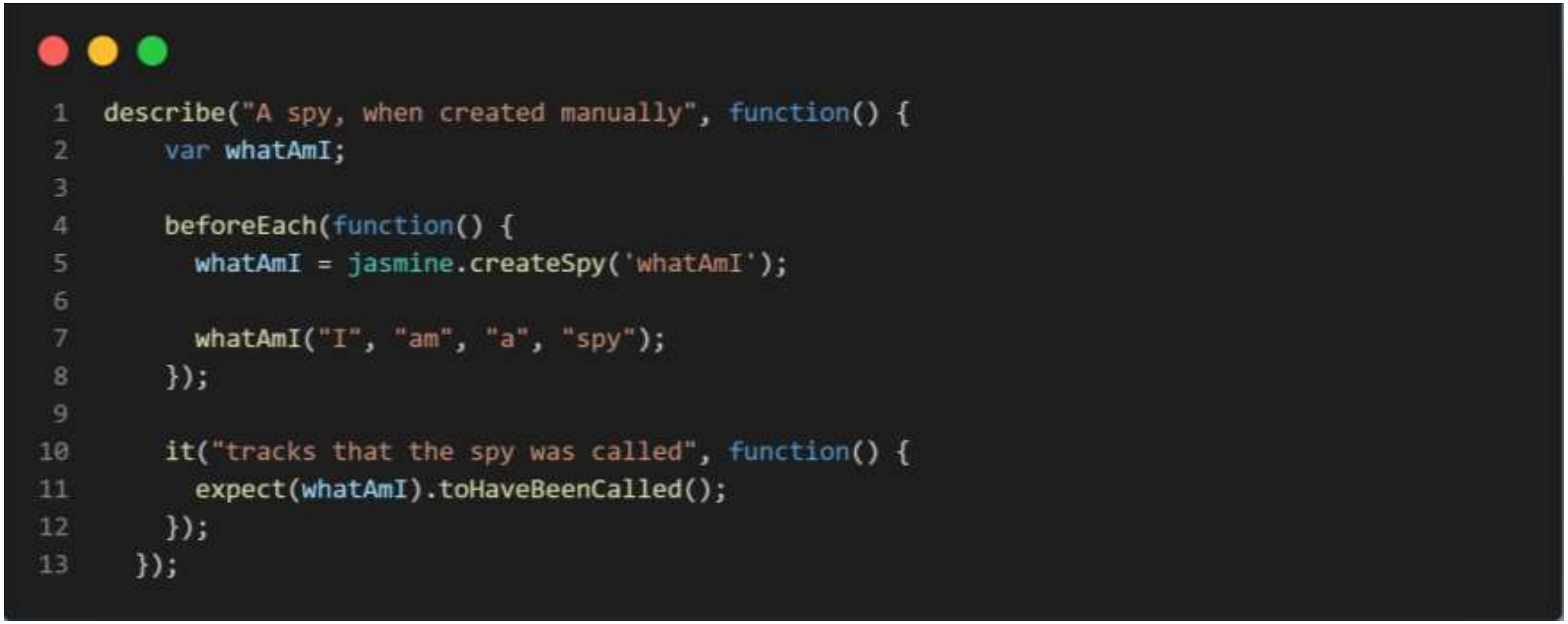
```
var foo,
   bar = null;

 beforeEach(function () {
   foo = {
     setBar: function (value) {
       bar = value;
     },
   };
   spyOn(foo, "setBar");

   foo.setBar(123);
   foo.setBar(456, "another param");
 });


 it("tracks that the spy was called", function () {
   expect(foo.setBar).toHaveBeenCalled();
 });
 it("tracks that the spy was called x times", function () {
   expect(foo.setBar).toHaveBeenCalledTimes(2);
 });
```

Spies: createSpy



```
describe("A spy, when created manually", function() {
    var whatAmI;

    beforeEach(function() {
      whatAmI = jasmine.createSpy('whatAmI');

      whatAmI("I", "am", "a", "spy");
    });

    it("tracks that the spy was called", function() {
      expect(whatAmI).toHaveBeenCalled();
    });
  });
```

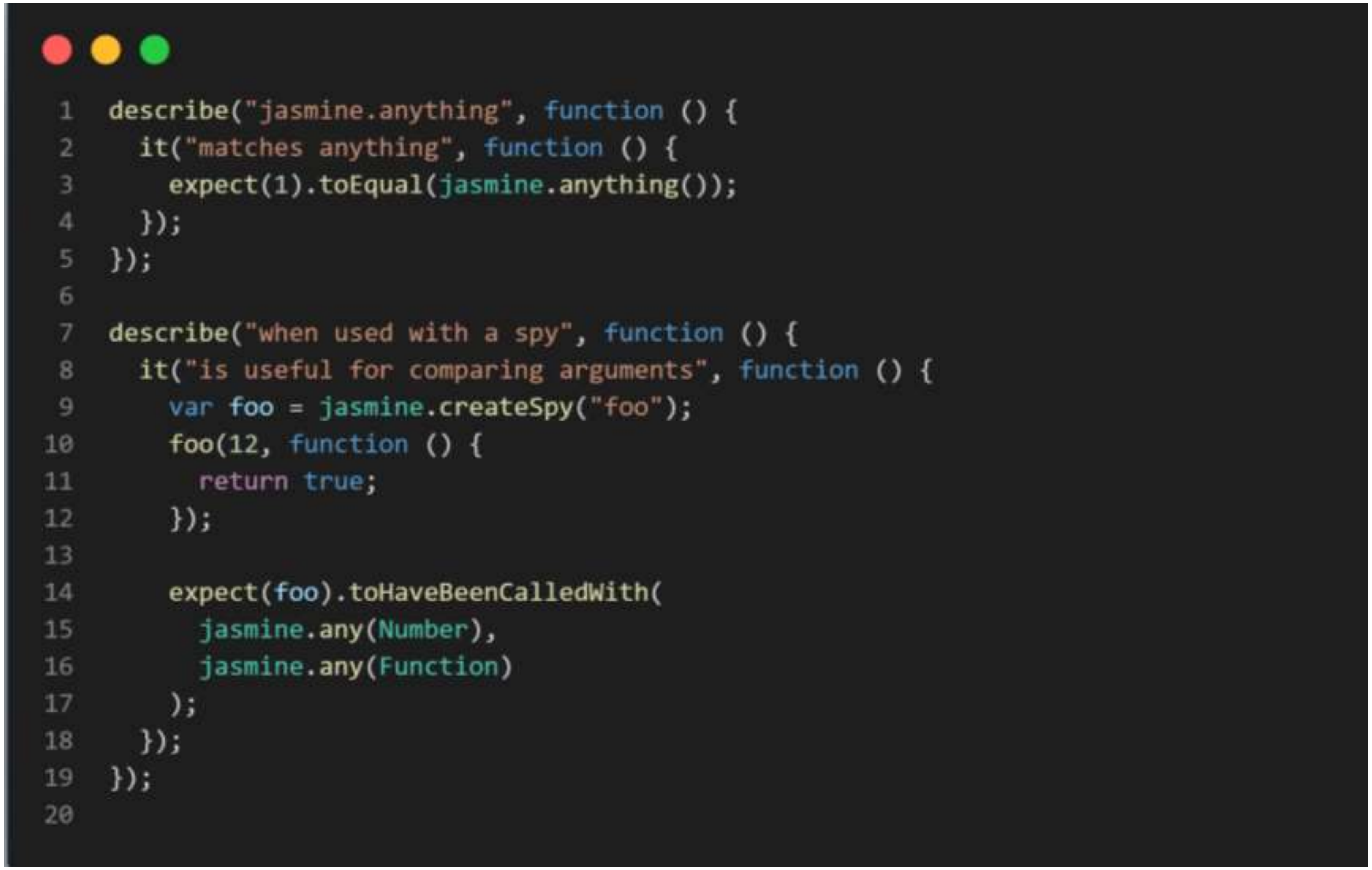
Any &Anything

Sometimes you don't want to match with exact equality. Jasmine provides a number of asymmetric equality testers.

*-Jasmine.any takes a constructor or "class" name as an expected value. It returns true if the constructor matches the constructor of the actual value.

*- jasmine.anything returns true if the actual value is not null or undefined.

Example :



```
describe("jasmine.anything", function () {
  it("matches anything", function () {
    expect(1).toEqual(jasmine.anything());
  });
});

describe("when used with a spy", function () {
  it("is useful for comparing arguments", function () {
    var foo = jasmine.createSpy("foo");
    foo(12, function () {
      return true;
    });

    expect(foo).toHaveBeenCalledWith(
      jasmine.any(Number),
      jasmine.any(Function)
    );
  });
});
```

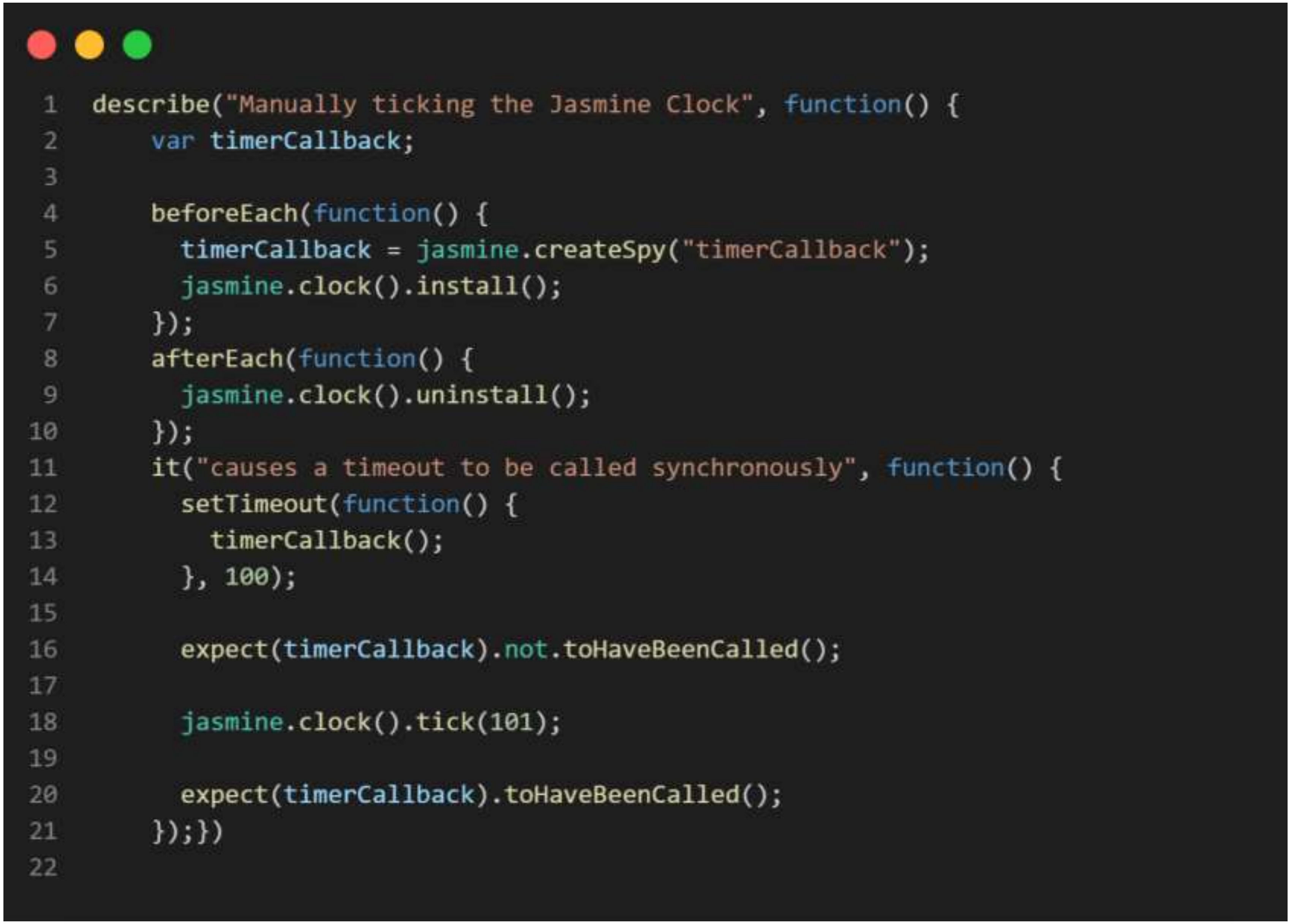
Jasmine Clock

The Jasmine Clock is available for testing time dependent code.

It is installed with a call to jasmine.clock().install in a spec or suite that needs to manipulate time.

Be sure to uninstall the clock after you are done to restore the original functions

Example :



```
describe("Manually ticking the Jasmine Clock", function() {
    var timerCallback;

    beforeEach(function() {
      timerCallback = jasmine.createSpy("timerCallback");
      jasmine.clock().install();
    });
    afterEach(function() {
      jasmine.clock().uninstall();
    });
    it("causes a timeout to be called synchronously", function() {
      setTimeout(function() {
        timerCallback();
      }, 100);

      expect(timerCallback).not.toHaveBeenCalled();

      jasmine.clock().tick(101);

      expect(timerCallback).toHaveBeenCalled();
    });})

```