Spying on Functions

Nathan Taylor
SOFTWARE ENGINEER

@taylonr taylonr.com

What Are Spies?

Test Spy

A function that **observes** a specified function

What a Spy Observes

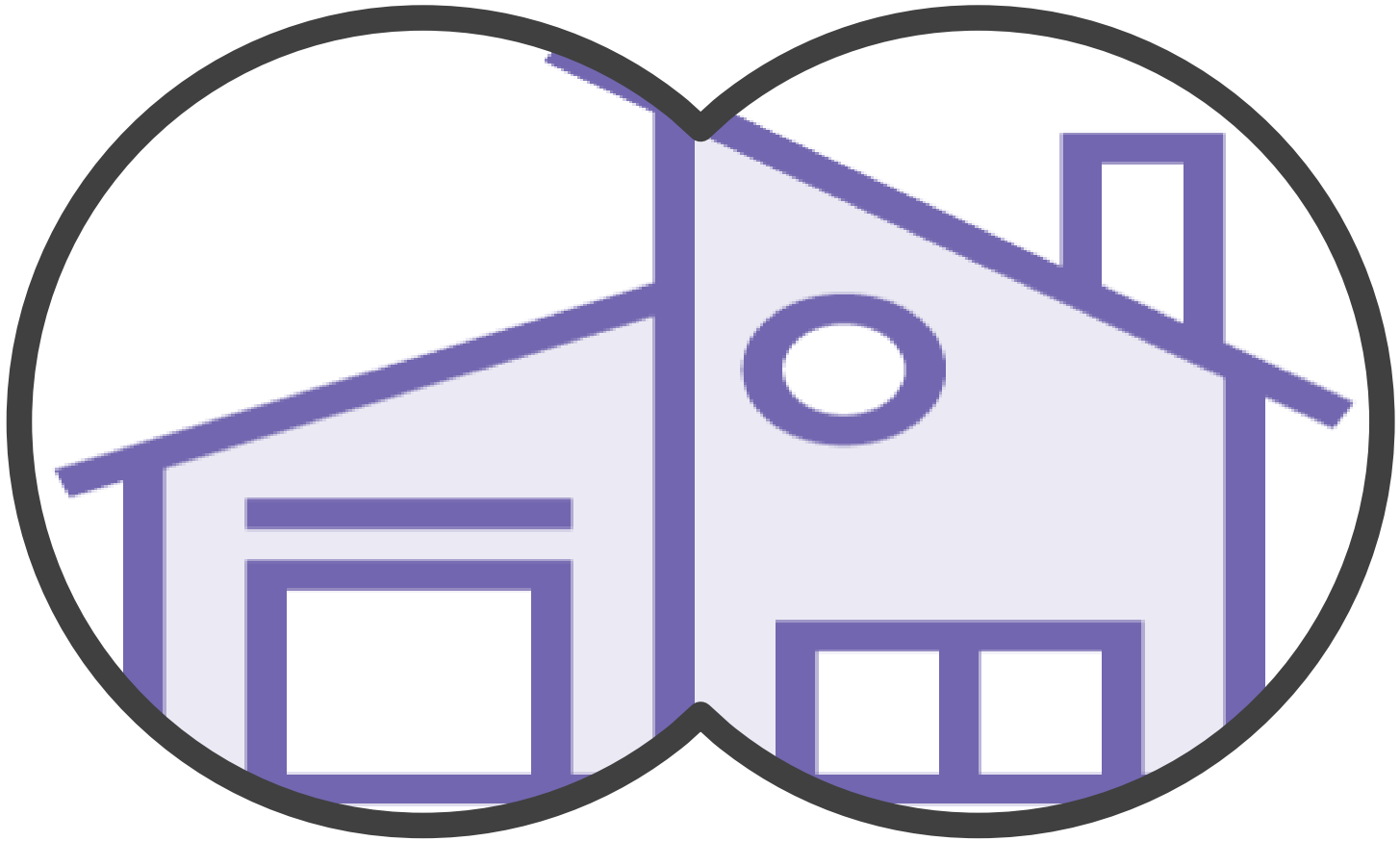
Number of times called

Parameters passed in

Return values

Exceptions thrown

Spies Are Like Nosey Neighbors



Anonymous Spy

```
const mySpy = sinon.spy();
```

JavaScript Callback

```
function alert(message, callback){
    callback(message);
}

const mySpy = sinon.spy();
alert("I'm watching a Sinon course on Pluralsight", mySpy);
```

Spying on Existing Functions

```
const Wishlist = {
  addItem: (item) => { return append(item, items); },
  removeItem: (item) => { return reject(item, items); }
}

const addItemSpy = sinon.spy(Wishlist, 'addItem');
```

Spied Function

```
{
  addItem: sinon.spy,
  removeItem: (item) => { return reject(item, items); }
}
```

Spy Details

```
{
  [Function: proxy]
  resetHistory: [Function: resetHistory],
  invoke: [Function: invoke],
  named: [Function: named],
  getCall: [Function: getCall],
  getCalls: [Function: getCalls],
…
}
```

Verifying a Function Was Called

Check That Spy Was Called

```
const mySpy = sinon.spy();
alert("I'm watching a Sinon course on Pluralsight", mySpy);


expect(mySpy.called).to.be.true;
```

```
expect(mySpy.called).to.be.true;
```

```
expect(mySpy.called).to.be.true;
```

```
expect(mySpy.called).to.be.true;
```

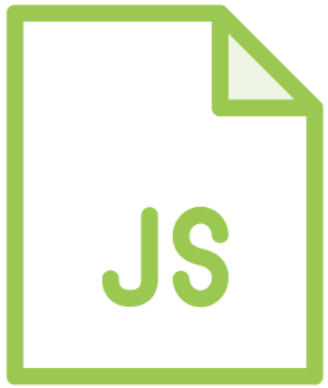
Check That Spy Was Called

```
const mySpy = sinon.spy();
alert("I'm watching a Sinon course on Pluralsight", mySpy);


expect(mySpy.called).to.be.true;
```

Demo: Verify Create Was Called

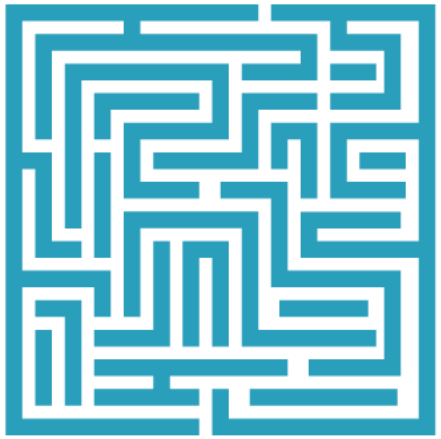
Testing Pro-tip



BookController.js

Crud.js

Write tests against crud.js



Common pattern on large projects

An Ordered Approach

Fix BookController

Move to Crud

Add new BookController

Verifying Multiple Calls

```
initializeNewUser = () => {
  db.all('preferences’)
    .then(...);

  db.all('history’)
    .then(...);


  db.all('products’)
    .then(...);
}
```

◂ Fetch user preferences

◂ Fetch order history

◂ Fetch available products

Testing Initialize New User

```
const get = sinon.spy(db, 'all');

initializeNewUser();

expect(db.all.called).to.be.true;
```

Application error between database calls

Verify Function Was Called Three Times

```
expect(db.all.calledThrice).to.be.true;
```

More Than Three Calls?

```
expect(db.all.callCount).to.equal(4);
```

Sinon can determine the number of times a function was called

Examining Parameters Passed to a Function

Verify Function Was Called Three Times

```
expect(db.all.callCount).to.equal(3);
```

spy.callCount

withArgs

```
sinon.spy(db, 'all');

expect(db.all.withArgs('preferences').calledOnce)
  .to.be.true;
```

Two failing tests

Matching Arguments

```
concat('Charles', 'Spurgeon');
concat('Charles', 'Wesley');

expect(concat.withArgs('Charles').called).to.be.true;
```

Checking Arguments

calledWith

calledOnceWith

alwaysCalledWith

calledWithExactly

Failing Test

```
concat('Charles', 'Spurgeon');
concat('Charles', 'Wesley’);

expect(concat.calledWithExactly('Charles')).to.be.true
```

Checking Arguments

calledWith

calledOnceWith

alwaysCalledWith

calledWithExactly

alwaysCalledWith

neverCalledWith

Failed Test Output

1) Books controller test:

AssertionError: expected false to be true

+ expected - actual

-false

+true

Verifying Create Works

```
expect(model.create.calledWith({ title: 'Test Book',
author: 'John Q Public',
publicationDate: '2018-01-01',
isbn: '1234'}).to.be.true
```

args

Two dimensional array. First dimension holds calls.
Second dimension holds parameters.

Testing Arguments

```
concat('Charles', 'Spurgeon');
concat('Charles', 'Wesley');

expect(concat.args[0][1]).to.equal('Spurgeon');
```

getCall

```
concat('Charles', 'Spurgeon');
concat('Charles', 'Wesley');

expect(concat.getCall(0).args[1]).to.equal('Spurgeon');
```

Better Failed Test Output

AssertionError: expected ‘Spurgeon’ to equal ’Jones’

+ expected - actual

-Spurgeon

+Jones

Demo: Verify the Right Book Was Created

Possible Test Failures

Undefined to equal ‘Test Book’

Cannot read property 'title' of undefined

Cannot read property ‘0’ of undefined

Verifying Exceptions

Function Throwing an Exception

```
addToList = (list, item) => {
  if(!list){
    throw new Error('List is not defined');
  }

  return list.concat(item);
}
```

threw()

```
const spy = sinon.spy(Wishlist, 'addToList');

try {
  Wishlist.addToList(null, {id: 1});
} catch(){}

expect(spy.threw()).to.be.true;
```

threw('TypeError')

threw(obj)

alwaysThrew(obj)

Spies as Building Blocks

Spy when you're not testing what a function does

Favor *args* over *called*

Stubbing out functions