Swapping a Stub for a Function

Nathan Taylor
SOFTWARE ENGINEER

@taylonr taylonr.com

What Is a Stub?

Stub

Test stubs are functions (spies) with pre-programmed behavior.

Stub

Test stubs are functions (spies) with pre-programmed behavior.

Stub

Test stubs are functions (spies) with pre-programmed behavior.

Stubs are like talking to a character

Stubs give YOU control

```
const myStub = sinon.stub();
```

◀ Anonymous stub

```
sinon.stub(Wishlist,
'addItem');
```

◀ Stub a specific function

```
sinon.stub(Wishlist)
```

◀ Stub ALL functions

sinon.stub(Wishlist)

```
{
    addItem: sinon.stub,
    removeItem: sinon.stub
}
```

Stub out individual functions

Why Stub?

```
try {
    functionWithRareException();
    db.save();
}
catch(err) {
}
```

Why Stub?

Control Flow

Prevent Undesirable Behavior

Returning a Fake Value

returns()

```
const pluralsight = sinon.stub().returns('Pluralsight');
```

```
.returns('Pluralsight');

.returns(['Spurgeon', 'Gill',
'Edwards']);

.returns({twitter:
'@taylonr'});

.returns(() => {});
```

- Single value
- Array
- Object
- Function

Matching Arguments

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

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

```
const myStub = sinon.stub();

myStub.withArgs('string').returns('Pluralsight');
myStub.withArgs('number').returns(1);

myStub('string'); //Pluralsight
myStub('number'); //1
myStub('bigNumber'); //undefined
```

withArgs removes unintended test data

onCall

```
myStub.onCall(0).returns(true);
myStub.onCall(1).returns(false);
myStub.returns("maybe");

myStub(); //true
myStub(); //false
myStub(); //maybe
myStub(); //maybe
```

Combine with fluent syntax

```
myStub.withArgs(1).onCall(1).returns(true);
myStub.returns(false);

myStub(); //false
myStub(2); //false
myStub(1); //false
myStub(1); //true
```

Handling Promises

```
sinon.stub(model, 'all').returns([{id: 1}]);
```

Stub model.all

Returns an array for every call

```
1) Books controller When getting a list of books Should
return 4 books:
     TypeError: model.all(...).then is not a function
      at Object.list (server/crud.js:55:8)
      at Context.it
(server/controllers/book.controller.spec.js:15:25)
```

Then Not Defined

Only returned an array

```
sinon.stub(model, 'all').returns(new Promise(() => {}));
```

Return a Promise

```
Error: Timeout of 2000ms exceeded. For async tests and
hooks, ensure "done()" is called; if returning a Promise,
ensure it resolves.
```

.resolves()

```
sinon.stub(Wishlist, 'addItem').resolves({id: 1});

Wishlist.addItem({}).then( data => {console.log(data)});
```

Interacting with Promises

.resolves()

.rejects()

```
sinon.stub(Wishlist,
'addItem').rejects();
```

```
sinon.stub(Wishlist,
'addItem').rejects({err:
'Fail!'});
```

- Rejects with **error**

- Rejects with the provided object

Chain with Promises

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

stub.withArgs({id: 1}).rejects({err: 'Already exists'});

stub.resolves({id: 2});
```

Demo: Stubbing Resolved Values

Cleaning up After a Stub

sinon.stub(Wishlist)

```
{
    addItem: sinon.stub,
    removeItem: sinon.stub
}
```

Multiple Stubs

```
it('Test 1', () => {
    sinon.stub(model, 'findById').rejects({});
    ...
});

it('Test 2', () => {
    sinon.stub(model, 'findById').resolves({});
    ...
});
```

```
sinon.stub(model, 'findById');
sinon.stub(model, 'findById');
```

Wrapping Stubs

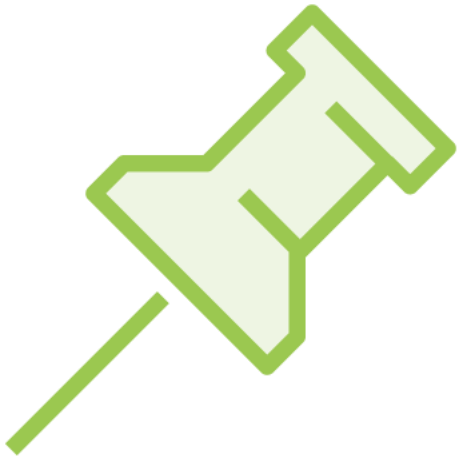
TypeError: Attempted to wrap findById which is already wrapped

```
sinon.stub(model, 'findById');
model.findById.restore();
```

.restore()

Resets the stubbed function back to its original state

Clean up stubs in `afterEach`



Clean up all stubs

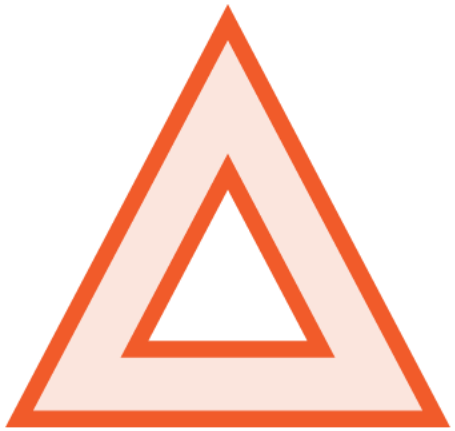
Additional Return Options

.throws()

```
sinon.stub(Wishlist, 'addItem').throws({err: {
    msg:'Database disconnected!'
}});
```

.callsFake()

```
const utils = { concat: (firstName, lastName) => {}};

sinon.stub(utils, 'concat’)
    .callsFake((firstName, lastName) => {
        return firstName + lastName
    });

utils.concat('Nate', 'Taylor'); //NateTaylor
```



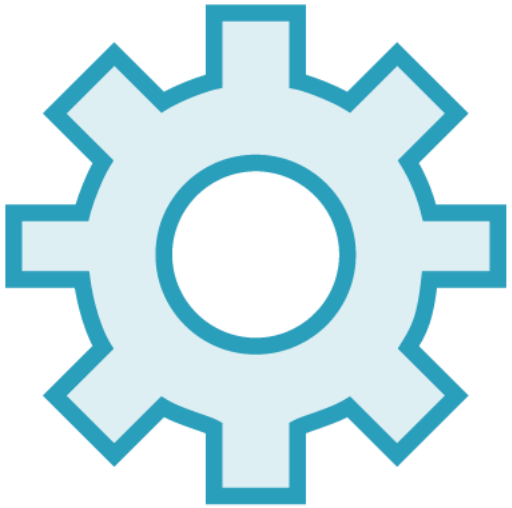
Your fake function might have an error

Extend SinonJS with .addBehavior()

. addBehavior()

```
sinon.addBehavior('log', (fake, msg) => {
console.trace(msg); });

sinon.stub(Wishlist, 'addItem').log('Called here');
```

```
Trace: Called Here

    at Object.sinon.addBehavior
(/Users/nate/code/pluralsight/sinonjs/server/controllers/book.controller.spec.js:42:59)

    at Object.proto.(anonymous function) [as trace]
(/Users/nate/code/pluralsight/sinonjs/node_modules/sinon/lib/sinon/behavior.js:221:12)

    at Function.trace
(/Users/nate/code/pluralsight/sinonjs/node_modules/sinon/lib/sinon/behavior.js:214:46)
```



Sinon gives you power

Spies vs. Stubs

Spies don’t alter functionality

Stubs aren’t checked with assertions

Why Stub?

Control Flow

Prevent Undesirable Behavior

Spies and stubs combined?