Spying With Sinon

Stubs, Spies and Mock Objects explained

Agenda

- Reasons to mock
- Vanilla mocking
- How sinon can help
- Stubs and Spies
- Faking timers
- Faking the server

Reasons To Mock

Reasons To Mock

$.ajax

PersonalData

setTimeout

Reasons To Mock

Reasons To Mock

- PersonalData object can save data to server
- If saving failed, it retries 3 times

Reasons To Mock

- Both server and clock are external
- We prefer to test in isolation

What We Can Do

- Provide our own $.ajax, that won't go to the server
- Provide our own setTimeout that won't wait for the time to pass

What We Can Do

- Lab: Given the class here https://gist.github.com/ynonp/6667146
- Write a test case to verify sendData actually retried 3 times

What We Can Do

- Solution: https://gist.github.com/ynonp/6667284


Mocking Notes

- Solution is far from perfect.
- After the test our “fake” methods remain
- Writing “fake” methods was not trivial

This Calls for Sinon


About Sinon

- A JS mocking library
- Helps create fake objects
- Helps tracking them

About Sinon

- Homepage:
  http://sinonjs.org/
- Google Group:
  http://groups.google.com/group/sinonjs
- IRC Channel:
  #sinon.js on freenode

Solving with Sinon

- Here's how sinon might help us with the previous task
- Code: https://gist.github.com/ynonp/6667378


Solution Notes

- Sinon's fake timer was easier to use than writing our own
- Now we have a synchronous test (instead of async)

Let's Talk About Sinon

Spies

- A spy is a function that provides the test code with info about how it was used

Spies Demo

```
describe('Sinon', function() {
  describe('spies', function() {

    it('should keep count', function() {

      var s = sinon.spy();
      s();
      assert.isTrue(s.calledOnce);

      s();
      assert.isTrue(s.calledTwice);

      s();
      assert.equal(s.callCount, 3);

    });
  });
});
```

Spy On Existing Funcs

```
describe('Sinon', function() {
  describe('spies', function() {

    it('should keep count', function() {
      var p = new PersonalData();
      var spy = sinon.spy(p, 'sendData');

      p.sendData();

      assert.isTrue( spy.calledOnce );
    });
  });
});
```

Spies Notes

- Full API:
  http://sinonjs.org/docs/#spies
- Tip: Use as callbacks

Spy + Action = Stub


Stub Demo

- Let's fix our starting example
- We'll replace $.ajax with a stub
- That stub always fails

Stub Demo

```
var stub = sinon.stub(jQuery, 'ajax').yieldsTo('error');

describe('Data', function() {
  describe('#sendData()', function() {

    it('should retry 3 times before quitting', function() {
      var p = new PersonalData();
      p.sendData();
      assert.equal(stub.callCount, 1);
    });
  });
});
```

What Can Stubs Do

```
var callback = sinon.stub();

callback.withArgs(42).returns(1);

callback.withArgs(1).throws("TypeError");
```

Stubs API

- Full Stubs API docs: http://sinonjs.org/docs/#stubs
- Main actions:
  - return stuff
  - throw stuff
  - call stuff

Spies Lab

- Given code here:
  https://gist.github.com/ynonp/7101081
- Fill in the blanks to make the tests pass


Fake Timers

- Use `sinon.useFakeTimers()` to create a fake timer
- Use `clock.restore()` to clear fake timers

Fake Timers

- Use `tick(...)` to advance
- Affected methods:
  - setTimeout, setInterval, clearTimeout, clearInterval
  - Date constructor

Fake Servers

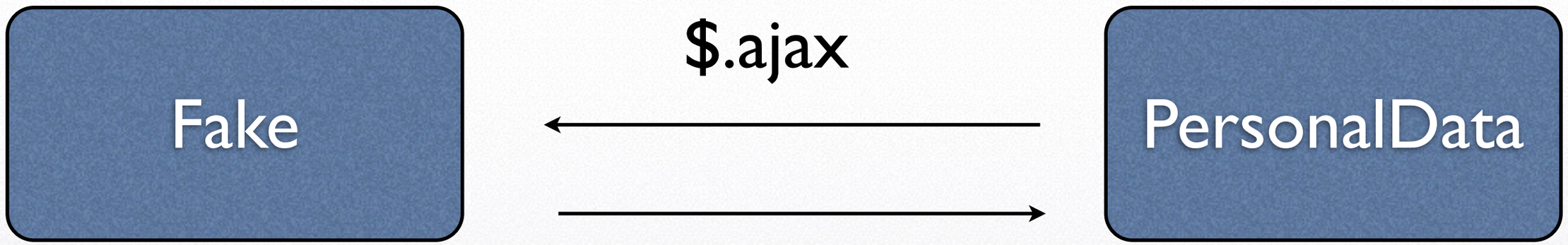
- Testing client/server communication is hard
- Use fake servers to simplify it

Fake Servers

$.ajax

PersonalData

Fake Servers




Let's write a test for the following class

```
1.  function Person(id) {
2.    var self = this;
3.
4.    self.load = function() {
5.      var url = '/users/' + id;
6.
7.      $.get('/users/' + id, function(info) {
8.        self.name = info.name;
9.        self.favorite_color = info.favorite_color;
10.     });
11.   };
12. }
```

Testing Plan

- Set-up a fake server
- Create a new Person
- call load()
- verify fields data

Setting Up The Server

```
1.  var server = sinon.fakeServer.create();
2.
3.  var headers  = {"Content-Type" : "application/json"};
4.  var response = JSON.stringify(
5.                    {"name" : "joe", "favorite_color": "blue" });
6.
7.  server.respondWith("GET", "/users/7",
8.                         [200, headers, response]);
9.  // now requesting /user/info.php returns joe's info as a JSON
```

Loading a Person

```
1.  var p = new Person(7);
2.  // sends a request
3.  p.load();
4.
5.  // now we have 1 pending request, let's fake the response
6.  server.respond();
```

Verifying the Data

```
1.  // finally, verify data
2.  expect(p.name).to.eq('joe');
3.  expect(p.favorite_color).to.eq('blue');
4.
5.  // and restore AJAX behavior
6.  server.restore();
```

Fake Server

- Use `respondWith()` to set up routes
- Use `respond()` to send the response

Fake Server

- Regexps are also supported, so this works:

```
1.  server.respondWith(/\/todo-items\/(\d+)/, function (xhr, id) {
2.      xhr.respond(
3.        200,
4.        { "Content-Type": "application/json" },
5.        '[{ "id": ' + id + ' }]');
6.  });
```

Fake Server

- For fine grained control, consider fake XMLHttpRequest
- http://sinonjs.org/docs/#server

Wrapping Up

Wrapping Up

- Unit tests work best in isolation
- Sinon will help you isolate units, by faking their dependencies


Wrapping Up

- Write many tests
- Each test verifies a small chunk of code
- Don’t test everything

Online Resources

- Chai: http://chaijs.com/
- Mocha: http://visionmedia.github.io/mocha/
- Sinon: http://sinonjs.org/
- Karma (test runner): http://karma-runner.github.io/0.10/index.html


Thanks For Listening

- Ynon Perek
- http://ynonperek.com
- ynon@ynonperek.com