JavaScript Unit Testing using Mocha

Speaker: NC Patro

Outline

Mocha deep dive

Asserts with Chai

Mocking with Sinon

Code Coverage with Istanbul

Testing

Unit Testing

Integration Testing

Functional Testing

What is Unit Testing

- ❏ Find the smallest part of an application, called units.
- ❏ Just test that unit to check whether it’s fit for use or not.
- ❏ Mock rest of the things which are not under that test.
- ❏ If we are going to create a test for any function, then we need to make sure that the function by itself, separate from everything, should do what it is intended to do, not more, not less. And that’s called a unit test.
- ❏ Everytime, we change the function, the unit test runs to make sure the functionality is fine.

Integration Testing

Testing the interactions between units.

Testing scenarios like Function calls another function with some context.

Mock rest of things like outside resources.

Mocha

Mocha is JavaScript test framework.

Runs on Node.js and Browser.

Official site: https://mochajs.org

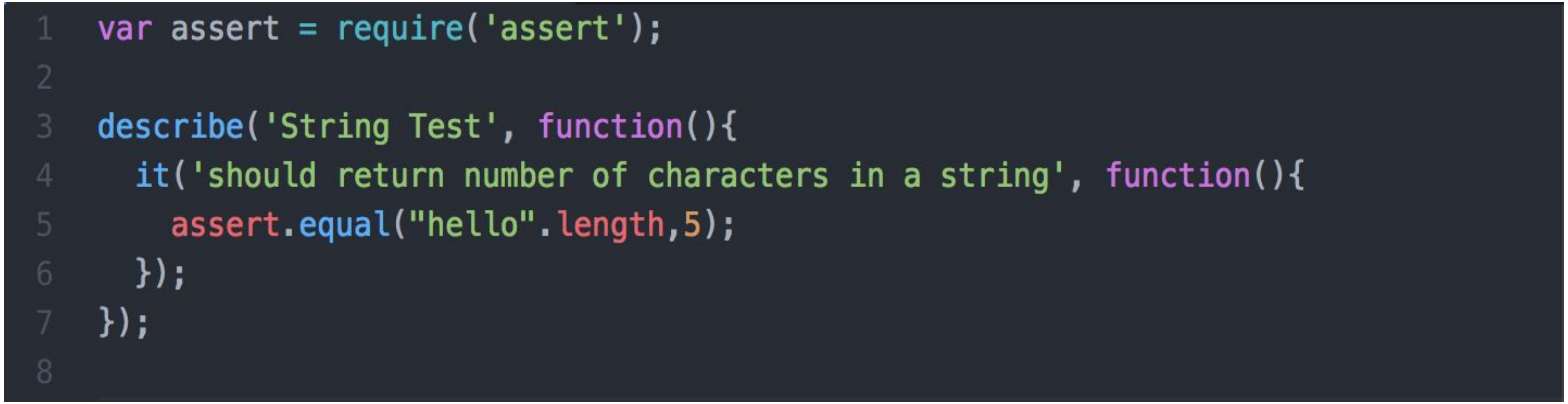
Installation:

```
$ npm install --global mocha
```

```
$ npm install --save-dev mocha
```

To install Mocha v3.0.0 or newer with npm, we need npm v2.14.2 or newer. Additionally, to run Mocha, we need Node.js v4 or newer.

Mocha Basic Spec



```
var assert = require('assert');

describe('String Test', function(){
  it('should return number of characters in a string', function(){
    assert.equal("hello".length,5);
  });
});
```

Test Assertion

- Assertion is an expression which helps system to know code under test failed.
- Assert just throws an error when things are not right.
- Assert tests the expression, it does nothing when it passes and throws exception to tell test framework about failure.
- We can just throw an exception to fail the test as well.

Testing actual code

- ❏ Testing a function
- ❏ Testing Async Function (callback)
- ❏ Handling Timeouts
- ❏ Hooks like beforeEach and afterEach
- ❏ Adding pending test
- ❏ Run only specific test or spec
- ❏ Skip test or spec

Assertion with Chai

Chai is BDD/TDD assertion library

Can be paired with any javascript testing framework.

Official site: http://chaijs.com

Installation:

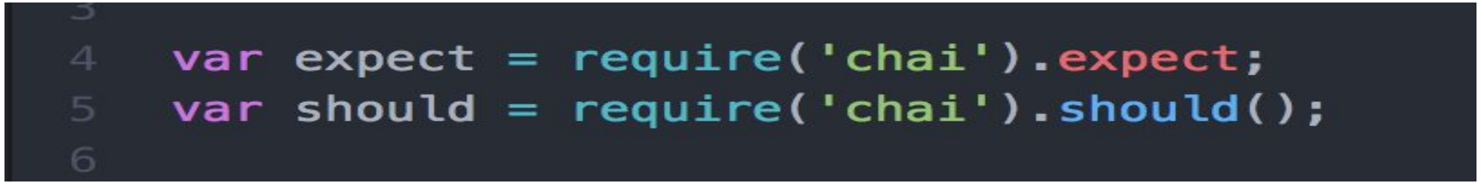
```
$ npm install chai --save
```

Assertion with Chai is more like natural language assertions.

Assertion interfaces or styles

Assertion with ***expect***

Assertion with ***should***



```
var expect = require('chai').expect;
var should = require('chai').should();
```

The expect interface provides function for assertion while should interface extends each object with a should property for assertion.

More assertion with should and expect

- ❏ Assertion with objects
- ❏ Assertion for Null
- ❏ Testing Async (Promises)

Mocking with Sinon

Standalone test spies, stubs and mocks for JavaScript

Works with any unit testing framework

Official site: http://sinonjs.org

Installation:

```
$ npm install sinon --save
```

Spy - sinon.spy()

A test spy is a fake function.

spy function helps to track the execution of function.

Spy function records arguments, return value, number of times called, any exception etc.

Spy the function - sinon.spy(user, ‘isAdmin’)

Sinon.spy() can watch the existing function execution as well.

First argument of Sinon.spy() is object name and second is function or method name which we want to watch.

Stub the function

Stub helps to replace the function

sinon.stub(obj, ‘function’)

We can return our own returns and exceptions.

Mocking the function

Mocks (and mock expectations) are fake methods (like spies) with pre-programmed behavior (like stubs) as well as pre-programmed expectations.

sinon.mock()

Code coverage with Istanbul

Istanbul is code coverage tool.

It generates few reports to whether we are testing enough or not.

Installation:

```
$ npm install -g istanbul
```

To generate coverage report

```
istanbul cover node_modules/.bin/_mocha -- test/**/*
```

Thank You