Ji Zhou
Net id: jz65
April 20, 2017

Javascript Unit Test Frameworks

Jasmine vs Mocha

Abstraction

There are many unit test frameworks out there for developers to choose for Javascript. But Jasmine and Mocha are the most popular. They have many aspects in common. And there are also differences between them. But to be honest, it is very similar to use these two frameworks.

Introduction

In our assignments, we used test-driven development process. It is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only. This is opposed to software development that allows software to be added that is not proven to meet requirements[1]. In test driven development cycle, we write a new test and then run all tests see if the new test fails and then write the unit code. It is done when the code passes the test. So mostly, we would like to have smaller modules or units to test. It will be much more helpful for us to design and separate concerns to smaller modules.

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use[2]. And I am going to talk about two unit test frameworks in Javascript: Jasmine and Mocha. The two share a lot of similarities.

API

For API, you just need to write test suite with "describe" blocks and using it function for each test. It all looks like following:

```
describe(‘funcX()…’, function(){
     it( ‘should do … ’, function() {
          //assertions
     })
})
```

Jasmine has its own built-in assertion library. Things are different for Mocha as Mocha does not have one and relies on third-party libraries. There are some options for nodejs and browser like Chai, should.js, expect.js and better-assert. And Chai is a popular choice of assertion library for those who uses Mocha. So if you choose to use Mocha, you will have to load the assertion library to your test utilities. For Chai, there are three types of assertion: expect, assert and should.

Jasmine `expect().toEqual()` Chai `expect().to.equal()`

It would be easy for developer who uses Jasmine to switch to Mocha.

Test Doubles

Test double is a generic term used for following objects or procedures. In object-oriented programming, developers would employ a technique called automated unit testing to enhance the quality of the software. Frequently, the final release software consists of a complex set of objects or procedures interacting together to create the final result. In automated unit testing, it may be necessary to use objects or procedures that look and behave like their release-intended counterparts, but are actually simplified versions that reduce the complexity and facilitate testing[3]. There are four basic types of test doubles.

First, test stub is used for incremental top-down testings. Stubs are modules that simulate the behavior of actual software modules that another module who is undergoing test depends on.

Second, mock object is simulated object that mimic the behavior of real objects for verifying the output of tested procedures.

Third, fake objects are mostly simpler implementations before release version, for example, using a in-memory database instead of accessing a real database. It records how a function is called or used.

Last, test spy which is used in Jasmine is also used for verifying the output of test part by asserting expectations, but without defining the expectations before test is executed.

However, Mocha does not come with a test double library. You may need to load Sinon into test harness. Sinon works well with most Javascript test frameworks. So even thou Jasmine has its own test double library, you can also combine it with Sinon.

Asynchronous Testing

In Jasmine 2.x and Mocha ,   asynchronous testing is the same. funca( ) uses fetch which performs the XHR request. And we may need to assert that funca( ) does successfully. So there is no real AJAX request is made here. We achieve asynchronous testing just by simply passing a parameter in the it callback function (in most situations we call it done but you can call it whatever you like). The test runner will pass in a function and wait for the function to be executed before test done. The test will throw error and time out if the function is not called within a certain time period.

```
it( ‘should do … ’, function(done) {
      funca().then(func(){
            … …
            done();
      })
```

Fake Server

Jasmine has not inbuilt fake server feature. Mocha can do it with Sinon. So you can setup fake responses to AJAX requests made for certain URLs. It allows developers to test code that makes AJAX requests regardless of which library is used. And you could do preprocessing on the response for a function that makes an AJAX call.

Running Tests

Mocha can be run using a command line utility. Jasmine do not have its own test runner. There are several test runners out there for Jasmine and the most popular one is Karma. Karma can also be used with Mocha.

Conclusion

To summarize, Jasmine framework and Mocha have more similarities than difference. Jasmine has most thing built in itself including assertions and test double libraries. But it is not fully equipped for example it does not have a test runner so developers have to use tools like Karma. But for Mocha, it has to be used with many third party libraries to be fully functional. So if you prefer to use a almost setup framework, Jasmine will be better than Mocha as it is almost settled. But if combination of different tools is more attractive to you, Mocha suits you more.

Reference

[1]: https://en.wikipedia.org/wiki/Test-driven_development

[2]: https://en.wikipedia.org/wiki/Unit_testing

[3]: https://en.wikipedia.org/wiki/Test_double

[4]: https://github.com/sinonjs/sinon