Google | طويق

Testing and Refactoring Node.js

أكاديمية طويق
TUWAIQ ACADEMY

وزارة الاتصالات وتقنية المعلومات
MINISTRY OF COMMUNICATIONS AND INFORMATION TECHNOLOGY

الاتحاد السعودي للأمن السيبراني والبرمجة والدرونز
SAUDI FEDERATION FOR CYBERSECURITY, PROGRAMMING & DRONES

مهارات المستقبل | stc

Node.js Testing Frameworks

What is Mocha?

- It’s a testing framework for JavaScript.
- It runs on Node.js and the browser.
- You can use any assertion library you wish(e.g. Chai, Should.js, etc)
- Mocha provides the hooks before(), after(), beforeEach(), and afterEach().
  - ➢ Preconditions and clean up.
- It makes it easy to test asynchronous code (Promises & Async/Await)
- Extensible reporting

Google 1000 طويق

Node.js Testing Frameworks

What is Chai?

- Chai is an assertion library for Node.js and the browser.
- Can be paired with any JavaScript testing framework. (e.g. Mocha)
- Assertion with Chai provides natural language assertions, expressive and readable style.

Google | طويق

Node.js Testing Frameworks

Chai assertion styles

Should

```
chai.should();

foo.should.be.a('string');
foo.should.equal('bar');
foo.should.have.lengthOf(3);
tea.should.have.property('flavors')
  .with.lengthOf(3);
```

Expect

```
var expect = chai.expect;

expect(foo).to.be.a('string');
expect(foo).to.equal('bar');
expect(foo).to.have.lengthOf(3);
expect(tea).to.have.property('flavors')
  .with.lengthOf(3);
```

Assert

```
var assert = chai.assert;

assert.typeOf(foo, 'string');
assert.equal(foo, 'bar');
assert.lengthOf(foo, 3)
assert.property(tea, 'flavors');
assert.lengthOf(tea.flavors, 3);
```

Node.js Testing Frameworks

1. `npm i mocha`

2. Update your scripts in package.json file

```
"scripts": {

  "test": "mocha"
},
```

3. Write the back end code for your express app

```
//Load express module with `require` directive
var express = require('express')
var app = express()

//Define request response in root URL (/)
app.get('/', function (req, res) {
  res.send('Hello World')
})

//Launch listening server on port 8080
app.listen(8080, function () {
  console.log('App listening on port 8080!')
})
```

Node.js Testing Frameworks

3. install the request module like this

```
npm install request chai –save-dev
```

4. Create a test folder with a app.test.js file inside the test folder,
   The code that should be in the ‘app.test.js’ file. This is unit test using Mocha and Chai.

5. Write the code to test your backend endpoints like this

```
let expect  = require('chai').expect;
var request = require('request');

it('Main page content', function(done) {
    request('http://localhost:8080' , function(error, response, body) {
        expect(body).to.equal('Hello World');
        done();
    });
});
```

- Use Chai to write the actual code that will be used to execute the test.
- Use the it() function to describe the actual test

Node.js Testing Frameworks

6. run `npm test`

7. See the output of your tests

Node.js Testing Frameworks

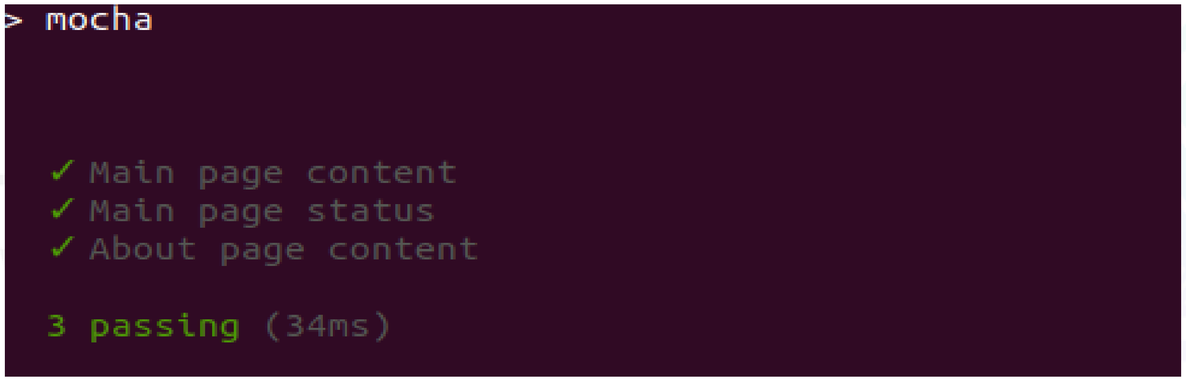
8. Write more test scripts in your test file to test different end points. See screenshot below on how to test for end points.

```
it('Main page status', function(done) {
    request('http://localhost:8080' , function(error, response, body) {
        expect(response.statusCode).to.equal(200);
        done();
    });
});

it('About page content', function(done) {
    request('http://localhost:8080/about' , function(error, response, body) {
        expect(response.statusCode).to.equal(404);
        done();
    });
});
```

Node.js Testing Frameworks

- Now run `npm test` to see the updated tests in your terminal.



- We can fail one test on purpose to see the different output, See example with about end point for this example ...

Node.js Testing Frameworks

- You can also group statements and tests together like this:

```
let expect  = require('chai').expect;
let request = require('request');
describe('Status and content', function() {
   describe ('Main page', function() {
       it('status', function(done){
           request('http://localhost:8080/', function(error, response, body)
{
               expect(response.statusCode).to.equal(200);
               done();
           });
       });

       it('content', function(done) {
           request('http://localhost:8080/' , function(error, response, body)
{
               expect(body).to.equal('Hello World');
               done();
           });
       });
   });

   describe ('About page', function() {
       it('status', function(done){
           request('http://localhost:8080/about', function(error, response,
body) {
               expect(response.statusCode).to.equal(404);
               done();
           });
       });

   });
});
```

Node.js Testing Frameworks

- The `assert()` module compares two values. If the two values aren’t equal, an error is thrown and the code execution is terminated.

```
let assert = require('assert');
assert(5 > 7);
```