## Talk Spec Backup

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

var TRUE = function() { /\*dummy\*/

};

describe("Talk", function() {

//Usualmente suelo hablar demasiado (y muy fuerte)

//esta vez quiero realizar otro enfoque .... más radical

it("no trata únicamente sobre TDD", TRUE);

it("trata sobre JS Agile Development", TRUE);

it("se centra sobre JS en el BROWSER - IMPORTANTE", TRUE); //AFTER 'Talk Spec: ejecuta los tests sobre NodeJS' . Talvez seguido del punto anterior.

describe("Talk Spec", function() {

it("explica los conceptos de la charla", TRUE);

it("utiliza Mocha", TRUE); //AFTER 'Mocha: funciona en el cliente y el servidor'

it("ejecuta los tests sobre NodeJS", TRUE); //Comenzamos a ejecutar los tests, Pensar si justo en este momento agregarmos los paquetes

});

describe("Demo", function() {

it("aplica los conceptos de la charla", TRUE);

it("implementa la funcionalidad del AHORCADO", TRUE); //Mostrar ejemplos de pantallas

it("utiliza Mocha y Chai como testing frameworks", TRUE); //AFTER 'Mocha: Permite utilizar cualquier librería de asertos'

it("comienza escribiendo el 1er feature test(high lvl test) ", TRUE);

});

//Comencemos!!!!!!!!

});

describe("Node.js Testing", function() { //AFTER 'Talk: se centra sobre JS en el Browser '. Ver donde ponerlo, mejor no hablar de eso

it("tiene similudes a JS en el Browser", TRUE);

it("presenta desafíos diferentes en comparación a JS sobre el Browser", TRUE);

});

describe("JS Testing", function() {

it("tiene muchas herramientas: QUnit, JSTestDriver, Mocha ...", TRUE);

describe("JSTestDriver", function() {

it("usa un servidor remoto que toma el control de un navegador", TRUE);

it("utiliza la consola para enviar los tests hasta el servidor remoto", TRUE);

it("ejecuta los tests en el servidor remoto ", TRUE);

it("devuelve los resultados al cliente", TRUE);

});

describe("Mocha", function() {

it("usa una sintaxis 'BDD' (similar a RSpec)", TRUE);

//Estamos utilizando Mocha en estos momentos

it("funciona en el cliente(Browser) y el servidor(Node.js)", TRUE); //opcional

it("no incluye un API de asertos", TRUE);

it("permite utilizar cualquier librería de asertos: Chai, Should ...", function() {

'Esto es un aserto utilizando Chai'.should.be.a('string');

});

it("necesita un archivo HTML para ejecutar los tests en el Browser", TRUE); //Generar un archivo HTML y explicarlo

});

});

## Primer Feature Test

* Crear el archivo feature\_spec.js
* Hablar sobre el framework y como agrupa y organiza los tests.
* Crear el sgte test

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | describe("Features", *function*() { | | 2 | it("should let enter the secret word", *function*() { | | 3 | *var* wordInput = $('<input>'); | | 4 | *var* acceptButton = $('<button>'); | | 5 | *var* wordOutput = $('<div>'); | | 6 | *var* game = new Game({ | | 7 | wordInput: wordInput, | | 8 | acceptButton: acceptButton, | | 9 | wordOutput: wordOutput | | 10 | }); | | 11 |  | | 12 | wordInput.val('cocodrillo'); | | 13 | acceptButton.click(); | | 14 | wordOutput.html().should.equal('\_\_\_\_\_\_\_\_\_\_'); | | 15 | }); | | 16 | }); | |  |

* Copiar y pegar el feature\_runner.html

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | <!DOCTYPE HTML> | | 2 | <html lang="en-US"> | | 3 | <head> | | 4 | <title>Test Runner Template</title> | | 5 | <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> | | 6 | <!-- MOCHA STYLE SHEET --> | | 7 | <link rel="stylesheet" href="../css/mocha.css"> | | 8 | </head> | | 9 | <body> | | 10 | <div id="mocha"></div> | | 11 | <!-- MOCHA LIBRARY--> | | 12 | <script type="text/javascript" src="../lib/mocha-1.4.2.js"></script> | | 13 | <!-- OTRAS LIBRERÍAS--> | | 14 | <script type="text/javascript" src="../lib/chai-1.2.0.js"></script> | | 15 | <script type="text/javascript" src="../lib/jquery-1.8.0.js"></script> | | 16 |  | | 17 | <!-- MOCHA y CHAI CONFIGURATION --> | | 18 | <script type="text/javascript"> | | 19 | /\* | | 20 | Mocha nos permite utilizar diferentes DSL(sintaxis): | | 21 | - BDD (Ejm: RSpec) | | 22 | - TDD (Ejm: JUnit) | | 23 | - QUnit (Similar a otra JS Testing Framework) | | 24 | \*/ | | 25 | mocha.setup('bdd'); | | 26 | chai.should(); | | 27 | </script> | | 28 |  | | 29 | <!-- TEST FILES (SPECS) --> | | 30 | <script type="text/javascript" src="feature\_spec.js"></script> | | 31 |  | | 32 | <!-- SRC FILES (SPECS) --> | | 33 |  | | 34 | <!-- EJECUTAR MOCHA --> | | 35 | <script type="text/javascript"> | | 36 | mocha.run(); | | 37 | </script> | | 38 | </body> | | 39 | </html> | |  |

* Ver que el Test ha fallado.
* Crear el archivo src/game.js y agregarlo al featurerunner.
* Implementar la clase hasta que se vea únicamente error del aserto.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | *function* Game(*attributes*) { | | 2 | this.init(attributes); | | 3 | } | | 4 |  | | 5 | *Game*.prototype = { | | 6 | init: *function*(*attributes*) { | | 7 | this.wordInput= attributes.wordInput; | | 8 | this.acceptButton= attributes.acceptButton; | | 9 | this.wordOutput=attributes.wordOutput; | | 10 | } | | 11 | }; | |  |

* Crear el archivo spec/game\_spec.js
* Copiar el archivo spec/feauture\_runner.html, pegarlo y renombrarlo a spec/unit\_runner.html.
  + Internamente remplazar el title por Unit Test Runner
  + Cambiar la referencia a feature\_spec.js por game\_spec.js
* Escribir el primer test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | describe("game", *function*() { | | 2 | it("should show format secret word with dashes", *function*() { | | 3 | *var* game=new Game({ | | 4 | wordinput: $('<input>'), | | 5 | acceptButton: $('<input>'), | | 6 | wordOutput: $('<div>') | | 7 | }); | | 8 |  | | 9 | *var* word = game.formatSecretWord('gato'); | | 10 |  | | 11 | word.should.equal('\_\_\_\_'); | | 12 | }); | | 13 | }); | |  |

* Hacer pasar el test.

|  |  |
| --- | --- |
| 5 | *Game*.prototype = { |
| 6 | init: *function*(*attributes*) { |
| 7 | this.wordInput = attributes.wordInput; |
| 8 | this.acceptButton = attributes.acceptButton; |
| 9 | this.wordOutput = attributes.wordOutput; |
| 10 | }, |
| 11 | formatSecretWord: *function*(*secretWord*) { |
| 12 | *var* word = ''; |
| 13 | for(*var* i = 0; i < secretWord.length; i++) { |
| 14 | word += '\_'; |
| 15 | } |
| 16 | return word; |
| 17 | } |
| 18 | }; |

* Crear el segundo test. (podemos crear un test muy similar al feature test pero más simple o solo completar el código para pasar directamente el feature test.

|  |  |
| --- | --- |
| 14 | it("should show the secret word", *function*() { |
| 15 | *var* wordInput = $('<input>'); |
| 16 | *var* acceptButton = $('<button>'); |
| 17 | *var* wordOutput = $('<div>'); |
| 18 | *var* game = new Game({ |
| 19 | wordInput: wordInput, |
| 20 | acceptButton: acceptButton, |
| 21 | wordOutput: wordOutput |
| 22 | }); |
| 23 |  |
| 24 | wordInput.val('a'); |
| 25 | acceptButton.click(); |
| 26 |  |
| 27 | wordOutput.html().should.equal('\_'); |
| 28 | }); |

* Hacemos pasar el test.
  + Primero creamos el onAccept.
  + Asignamos el evento sin proxy
  + Cambiamos por proxy.

|  |  |
| --- | --- |
| 5 | *Game*.prototype = { |
| 6 | init: *function*(*attributes*) { |
| 7 | this.wordInput = attributes.wordInput; |
| 8 | this.acceptButton = attributes.acceptButton; |
| 9 | this.wordOutput = attributes.wordOutput; |
| 10 | this.acceptButton.click($.proxy(this.onAcceptButtonClick, this)); |
| 11 | },  ……… |
| 19 | onAcceptButtonClick: *function*() { |
| 20 | *var* secretWord = this.wordInput.val(); |
| 21 | *var* formatedSecretWord = this.formatSecretWord(secretWord); |
| 22 | this.wordOutput.html(formatedSecretWord); |
| 23 | } |
| 24 | }; |

* Refactorizamos los unit tests.

|  |  |
| --- | --- |
| 1 | describe("game", *function*() { |
| 2 | *var* wordInput; |
| 3 | *var* acceptButton; |
| 4 | *var* wordOutput; |
| 5 | *var* game; |
| 6 |  |
| 7 | beforeEach(*function*() { |
| 8 | wordInput = $('<input>'); |
| 9 | acceptButton = $('<button>'); |
| 10 | wordOutput = $('<div>'); |
| 11 | game = new Game({ |
| 12 | wordInput: wordInput, |
| 13 | acceptButton: acceptButton, |
| 14 | wordOutput: wordOutput |
| 15 | }); |
| 16 | }); |
| 17 |  |
| 18 | it("should format the secret word with dashes", *function*() { |
| 19 | *var* word = game.formatSecretWord('gato'); |
| 20 |  |
| 21 | word.should.equal('\_\_\_\_'); |
| 22 | }); |
| 23 |  |
| 24 | it("should show the secret word", *function*() { |
| 25 | wordInput.val('a'); |
| 26 | acceptButton.click(); |
| 27 |  |
| 28 | wordOutput.html().should.equal('\_'); |
| 29 | }); |
| 30 | }); |