## 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

});

});

## Template Runner Inicial

<!DOCTYPE HTML>

<html lang="en-US">

<head>

<title>Test Runner Template</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<!-- MOCHA STYLE SHEET -->

<link rel="stylesheet" href="../css/mocha.css">

</head>

<body>

<div id="mocha"></div>

<!-- MOCHA LIBRARY-->

<script type="text/javascript" src="../lib/mocha-1.4.2.js"></script>

<!-- OTRAS LIBRERÍAS-->

<script type="text/javascript" src="../lib/chai-1.2.0.js"></script>

<script type="text/javascript" src="../lib/jquery-1.8.0.js"></script>

<!-- MOCHA y CHAI CONFIGURATION -->

<script type="text/javascript">

/\*

Mocha nos permite utilizar diferentes DSL(sintaxis):

- BDD (Ejm: RSpec)

- TDD (Ejm: JUnit)

- QUnit (Similar a otra JS Testing Framework)

\*/

mocha.setup('bdd');

chai.should();

</script>

<!-- TEST FILES (SPECS) -->

<script type="text/javascript" src="feature\_spec.js"></script>

<!-- SRC FILES (SPECS) -->

<!-- EJECUTAR MOCHA -->

<script type="text/javascript">

mocha.run();

</script>

</body>

</html>

* Ver que el Test ha fallado.
* Crear la clase game.js y el spec game\_spec.js

Nota: Decir a los asistentes que en esta oportunidad voy a hacer pasos más grandes

* Escribir el primer test y hacerlo pasar.
* Escribir el segundo test y hacerlo pasar.

describe("game", function() {

var wordInput;

var acceptButton;

var wordOutput;

var game;

beforeEach(function() {

wordInput = $('<input />');

acceptButton = $('<button />');

wordOutput = $('<div />');

game = new Game({

wordInput: wordInput,

acceptButton: acceptButton,

wordOutput: wordOutput

});

});

it("should let enter the secret word", function() {

wordInput.val('gato');

acceptButton.click();

game.secretWord.should.equal('gato');

});

it("should show the secret word with dashes", function() {

game.secretWord = 'gato';

var secretword = game.showFormatedSecretWord();

wordOutput.html().should.equal('\_\_\_\_');

});

});

* El código que pasa dicho tests

function Game(attributes) {

this.init(attributes);

}

Game.prototype = {

init: function(attributes) {

this.wordInput= attributes.wordInput;

this.acceptButton= attributes.acceptButton;

this.wordOutput=attributes.wordOutput;

this.acceptButton.click($.proxy(this.onAcceptButtonClick,this));

},

onAcceptButtonClick:function() {

this.secretWord=this.wordInput.val();

},

showFormatedSecretWord: function() {

var word = '';

for (var i = 0; i < this.secretWord.length; i++) {f

word += '\_';

}

this.wordOutput.html(word);

}

};