# Preparativos

* Abrir en el navegador.
  + Mocha (página principal y github)
  + Chai (página principal y github)
  + JSTestDriver
  + Testem
* Descargar el base de Github
* Abrir el directorio del ejercicio.
* Cambiar de rama a Try

# Explicación del proyecto

Este es el proyecto base donde implementaremos nuestra aplicación de ejemplo, solo tiene los directorios y archivos que necesitamos para comenzar, este proyecto también lo podemos descargar de Github.

Tenemos el directorio public donde se encontrarán todos los archivos estáticos de nuestra aplicación web. Tests, los tests que crearemos para nuestra aplicación.

Como dijimos anteriormente solo nos vamos a enfocar en JS a nivel de cliente, pero necesitamos una tecnología a nivel de servidor que nos permita publicar nuestros archivos.

En el directorio copypaste tengo algunos archivos ya preparados para agilizar la sesión.

# Explicación de como vamos a realizar el ejemplo.

En el archivo features.txt tenemos las features que necesitamos implementar en nuestro juego. Entonces vamos a realizar aplicar acceptance testing, para lo cuál crearemos primero un feature tests de manera muy sencilla, un tests de alto nivel que describa el funcionamiento de la aplicación en términos del usuario y aplicaremos varios ciclos de tdd hasta que nuestro tests pase.

# Primer Feature Test

* Crear el archivo feature\_spec.js y copiar la lista de features.
* Explicar qué es Mocha, sus características, ventajas y desventajas.
* XQ estamos utilizando Mocha?
* Mocha organiza nuestros tests dentro de bloques describe.

## Feature Test

* Crear el test.   
  (Empezamos escribiendo los elementos y luego la interacción y el assert, ya luego la definición de la clase Game)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 8 | describe("Features", *function*(){ | | 9 | it("should let enter the secret word", *function*(){ | | 10 | *var* wordInput=$('<input>'); | | 11 | *var* wordForm=$('<form>'); | | 12 | *var* wordDisplayed=$('<div>'); | | 13 |  | | 14 | *var* game=new Game({ | | 15 | wordInput:wordInput, | | 16 | wordForm:wordForm, | | 17 | wordDisplayed:wordDisplayed | | 18 | }); | | 19 | game.start(); | | 20 |  | | 21 | wordInput.val('cocodrillo'); | | 22 | wordForm.submit(*function*(e) {e.preventDefaul();}); | | 23 |  | | 24 | wordDisplayed.html().should.be.equal('\_\_\_\_\_\_\_\_\_\_'); | | 25 | }); | | 26 | }); | |

* Copiar el Features.html y explicarlo.
* Reorganizar la ventana.
* Lograr que se vea el asert
  + Crear el archivo src/game.js y agregarlo al featurerunner.
  + Crear el método start();
  + Agregar el wordForm.submit(function(e) {e.preventDefaul();}); para permitir que se muestre el aserto.

## Test Driven Development

* Crear la clase game\_spec.js
* Crear el primer tests

|  |
| --- |
|  |
| |  |  | | --- | --- | | 2 | describe("show the secret word with dashes", *function*() { | | 3 | it("should show a word with one character", *function*() { | | 4 | *var* wordInput=$('<input>'); | | 5 | *var* wordDisplayed=$('<div>'); | | 6 | *var* game = new Game({ | | 7 | wordInput:wordInput, | | 8 | wordDisplayed:wordDisplayed | | 9 | }); | | 10 | wordInput.val('g'); | | 11 |  | | 12 | game.showInitialWord(); | | 13 |  | | 14 | wordDisplayed.html().should.equal('\_'); | | 15 | }); | | 29 | }); | |

* Duplicar el feautures.html y renombrarlo unit.html.
  + Internamente remplazar el title por Unit Test Runner
  + Cambiar la referencia a feature\_spec.js por game\_spec.js
* Implementar el test de manera muy sencilla.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | *function* Game(*attributes*) { | | 2 | this.wordDisplayed= attributes.wordDisplayed; | | 3 | } | | 4 |  | | 5 | *Game*.prototype = { | | 6 | showInitialWord: *function*() { | | 7 | return this.wordDisplayed.html('\_'); | | 8 | } | | 9 | }; | |

* Crear el sgte test.

|  |
| --- |
|  |
| |  |  | | --- | --- | | 16 | it("should show a word with multiple characters", *function*() { | | 17 | *var* wordInput=$('<input>'); | | 18 | *var* wordDisplayed=$('<div>'); | | 19 | *var* game = new Game({ | | 20 | wordInput:wordInput, | | 21 | wordDisplayed:wordDisplayed | | 22 | }); | | 23 | wordInput.val('gato'); | | 24 |  | | 25 | game.showInitialWord(); | | 26 |  | | 27 | wordDisplayed.html().should.equal('\_\_\_\_'); | | 28 | }); | |

* Pasar el tests.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | *function* Game(*attributes*) { | | 2 | this.wordDisplayed= attributes.wordDisplayed; | | 3 | this.wordInput=attributes.wordInput; | | 4 | } | | 5 |  | | 6 | *Game*.prototype = { | | 7 | showInitialWord: *function*() { | | 8 | *var* word=''; | | 9 | for (*var* i = 0; i < this.wordInput.val().length; i++) { | | 10 | word=word+"\_"; | | 11 | } | | 12 | this.wordDisplayed.html(word); | | 13 | } | | 14 | }; | |
|  |

* Refactorizamos los tests.

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

* Tenemos que lograr que el showinitialword se llame al realizar un click en el botton
  + Implementarlo a mano y verificar el feature test.
  + Crear un tests donde probemos esto.
* Optamos por la segunda alternativa.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 30 | it("should show the initial word after click", *function*(){ | | 38 | submitButton.click(); | | 39 |  | | 40 | game.showInitialWord.should.be.called; | | 41 | }); | |

* Lamentablemente las frameworks de pruebas unitarias no nos ofrecen esta verificación de caja y necesitamos aplicar alguna técnica de mocking, en esta oportunidad vamos a utilizar la primera técnica de mocking que tenemos en JS, q es aprovechar las propias características del lenguaje.
* Pasamos el test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | *function* Game(*attributes*) { | | 2 | this.wordDisplayed= attributes.wordDisplayed; | | 3 | this.wordInput=attributes.wordInput; | | 4 | attributes.submitButton.click(this.onSubmitButtonClick); | | 5 | } | | 6 |  | | 7 | *Game*.prototype = { | | 8 | onSubmitButtonClick:*function*() { | | 9 | this.showInitialWord(); | | 10 | }, | | 18 | }; | |

* El test falla xq this tiene el valor del elemento sobre el cual se ha realizado un click.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 4 | attributes.submitButton.click($.proxy(this.onSubmitButtonClick,this)); | |

# Code Inspection JSHINT

* Cuando desarrollamos algo necesitamos verificar si nuestro código cumple determinadas convenciones o tiene errores potenciales, para esto utilizamos herramientas de análisis de código. Debemos ejecutar este análisis también de manera muy constante.
* Vamos a utilizar JSHint.
* Instalamos JSHint utilizando el gestor de paquetes de node.

Npm install –g jshint

* Agregamos los archivos .jshintrc.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | { | | 2 | // The Good Part && Style Preferences | | 3 | "curly": true, | | 4 | "eqeqeq": true, | | 5 | "immed": true, | | 6 | "latedef": true, | | 7 | "newcap": true, | | 8 | "noarg": true, | | 9 | "sub": true, | | 10 | "undef": true, | | 11 | "boss": true, | | 12 | "eqnull": true, | | 13 | "trailing": true, | | 14 | "noempty": true, | | 15 |  | | 16 | //Predefined Globals | | 17 | "browser": true, | | 18 | "jquery": true, | | 19 | "globals": { | | 20 | "afterEach": false, | | 21 | "beforeEach": false, | | 22 | "describe": false, | | 23 | "expect": false, | | 24 | "it": false, | | 25 | "fixtures": false | | 26 | } | | 27 | } | |

* Agregamos el archivo .jshintignore

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1 | tests/lib | |

* Para obtener los resultados.

jshint.cmd public/app/ tests --jslint-reporter > jshint.xml

jshint.cmd public/app/ tests --reporter tests/lib/junit-xml-formatter.js > jshint.xml

* Corregir los errores

*/\*global Game:false\*/*

## Utilizando GRUNT

* Instalación

Npm install –g grunt

* Creación de grunt.js

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | /\*global module:false\*/ | | 2 | *module*.exports = *function*(*grunt*) { | | 3 |  | | 4 | // Build configuration. | | 5 | grunt.initConfig({ | | 6 | lint: { | | 7 | files: ['grunt.js', 'src/\*.js', 'spec/\*.js'] | | 8 | }, | | 9 | jshint: { | | 10 | options: { | | 11 | curly: true, | | 12 | eqeqeq: true, | | 13 | immed: true, | | 14 | latedef: true, | | 15 | newcap: true, | | 16 | noarg: true, | | 17 | sub: true, | | 18 | undef: true, | | 19 | boss: true, | | 20 | eqnull: true, | | 21 | browser: true, | | 22 | jquery: true, | | 23 | trailing:true, | | 24 | noempty:true | | 25 | // strict:true, | | 26 | }, | | 27 | globals: { | | 28 | afterEach: false, | | 29 | beforeEach: false, | | 30 | describe: false, | | 31 | expect: false, | | 32 | it: false | | 33 | } | | 34 | } | | 35 | }); | | 36 |  | | 37 | // Default task. | | 38 | grunt.registerTask('default', 'lint'); | | 39 | }; | |

* Ejecutar grunt.
* Corregir los errores
  + Al inicio de las specs agregar: */\*global Game:false\*/*
  + Método formatSecretWord (agregarle paréntesis):  
     *for(var i = 0; i < secretWord.length; i++) { word += '\_'; }*

# Segundo Feature Test

* Refactorizamos los feature tests

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 9 | *var* wordInput; | | 10 | *var* wordDisplayed; | | 11 | *var* submitButton; | | 13 | *var* game; | | 14 |  | | 15 | beforeEach(*function*() { | | 16 | wordInput = $('<input>'); | | 17 | wordDisplayed = $('<div>'); | | 18 | submitButton = $('<button>'); | | 20 | game = new Game({ | | 21 | wordInput: wordInput, | | 22 | wordDisplayed: wordDisplayed, | | 23 | submitButton: submitButton, | | 25 | }); | | 26 | }); | |

## Feature Tests

* Creamos el segundo Test

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 33 | it("should be able to play", *function*() { | | 34 | wordInput.val('cocodrillo'); | | 35 | submitButton.click(); | | 36 |  | | 37 | $('button:contains("o")', buttonList).click(); | | 38 | $('button:contains("d")', buttonList).click(); | | 39 |  | | 40 | wordDisplayed.html().should.be.equal('\_o\_od\_\_\_\_o'); | | 41 | }); | |

* Agregamos el elemento buttonList en el beforeeach.

|  |  |
| --- | --- |
| 2 | *var* wordInput; |
| 3 | *var* wordForm; |
| 4 | *var* wordOutput; |
| 5 | *var* buttonList; |
| 6 | *var* game; |
| 7 |  |
| 8 | beforeEach(*function*() { |
| 9 | wordInput = $('<input>'); |
| 10 | wordForm = $('<button>'); |
| 11 | wordOutput = $('<div>'); |
| 12 | game = new Game({ |
| 13 | wordInput: wordInput, |
| 14 | wordForm: wordForm, |
| 15 | wordOutput: wordOutput, |
| 16 | buttonList: buttonList |
| 17 | }); |
| 18 | }); |

* Vemos el test fallar.

## Test Drive Development

* **Creamos tests de CREACIÓN DE BOTONES dentro de un nuevo describe.**

|  |  |
| --- | --- |
| 34 | describe("create letter buttons", *function*() { |
| 35 | it("should show the letter a", *function*() { |
| 36 | wordInput.val('gato'); |
| 37 | wordForm.submit(); |
| 38 |  |
| 39 | $('button:contains("a")', buttonList).should.have.length(1); |
| 40 | }); |
| 41 | }); |

* Hacemos pasar el test.

|  |  |
| --- | --- |
| 5 | *Game*.prototype = { |
| 6 | init: *function*(*attributes*) { |
| 7 | this.wordInput = attributes.wordInput; |
| 8 | this.wordForm = attributes.wordForm; |
| 9 | this.wordOutput = attributes.wordOutput; |
| 10 | this.buttonList = attributes.buttonList; |
| 11 | this.wordForm.click($.proxy(this.onFormSubmit, this)); |
| 12 | },  ……… |
| 20 | onFormSubmit: *function*() { |
| 21 | *var* secretWord = this.wordInput.val(); |
| 22 | *var* formatedSecretWord = this.formatSecretWord(secretWord); |
| 23 | this.wordOutput.html(formatedSecretWord); |
| 24 | this.createButtons(); |
| 25 | }, |
| 26 | createButtons: *function*() { |
| 27 | *var* buttonA='<button>a</button>'; |
| 28 | this.buttonList.append(buttonA); |
| 29 | } |
| 30 | }; |

* Escribimos un nuevo test.

|  |  |
| --- | --- |
| 42 | it("should show the letter z", *function*() { |
| 43 | wordInput.val('gato'); |
| 44 | wordForm.submit(); |
| 45 |  |
| 46 | $('button:contains("z")', buttonList).should.have.length(1); |
| 47 | }); |

* Hacemos pasar el test.

|  |  |
| --- | --- |
| 26 | createButtons: *function*() { |
| 27 | *var* buttonA='<button>'+*String*.fromCharCode(97)+'</button>'; |
| 28 | this.buttonList.append(buttonA); |
| 29 |  |
| 30 | *var* buttonZ='<button>'+*String*.fromCharCode(122)+'</button>'; |
| 31 | this.buttonList.append(buttonZ); |
| 32 | } |

* Escribimos un nuevo test.

|  |  |
| --- | --- |
| 49 | it("should show the letter b", *function*() { |
| 50 | wordInput.val('gato'); |
| 51 | wordForm.submit(); |
| 52 |  |
| 53 | $('button', buttonList).should.have.length(26); |
| 54 | }); |

* Hacemos pasar el test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 26 | createButtons: *function*() { | | 27 | for(*var* i = 97; i <= 122; i++) { | | 28 | *var* button = '<button>' + *String*.fromCharCode(i) + '</button>'; | | 29 | this.buttonList.append(button); | | 30 | } | | 31 | } | |  |

## Continuous Testing

* Se habrán dado cuenta que estar ejecutando constamente el navegador es un molestoso y quita tiempo. No hay una forma de poder ejecutarlos a través de la consola o automáticamente cuando detectan algún cambio en los archivos, de manera similar a lenguajes como C#(autotest.net) o java(infinitests)????
* En JS existen los tests runners, cuyo objetivo no es ofrecer un API para realizar pruebas, sino son herramientas que buscan facilitar y mejorar la ejecución de pruebas.
* JsTestDriver : Creo el modelo del cuál las demás herramientas se basan.
* Instalamos testem

Npm i testem - g

* Podemos enviar archivos de manera individual o aprovechar el nuestro runner.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1 | test\_page: spec/unit\_runner.html | |

* El requisito que debe tener nuestro runner es agregar un nuevo script. Tiene que estar debajo de mocha

<script type="text/javascript" src="/testem.js"></script>

* Esto nos permite todavía ejecutar nuestros runner en el navegador sin la necesidad de tener el servidor de testem levantado.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 13 | <script> | | 14 | if (location.hash === '#testem') | | 15 | *document*.write('<script src="/testem.js"></'+'script>') | | 16 | </script> | |

## Continuamos con Test Driven Development

* **Creamos los test de JUGAR CON LETRAS.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 57 | describe("try letters", *function*() | | 58 | it("should show the letter when the letter match", *function*() { | | 59 | wordInput.val("g"); | | 60 | wordForm.submit(); | | 61 |  | | 62 | game.tryLetter("g"); | | 63 | wordOutput.html().should.equal('g'); | | 64 | }); | | 65 | }); | |

* Hacemos pasar el test.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 32 | tryLetter: *function*(*letter*) { | | 33 | this.wordOutput.html(letter); | | 34 | } | |

* Creamos el siguiente test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 66 | it("should show underscore when the letter does not match", *function*() { | | 67 | wordInput.val("g"); | | 68 | wordForm.submit(); | | 69 |  | | 70 | game.tryLetter("e"); | | 71 | wordOutput.html().should.equal('\_'); | | 72 | }); | |

* Hacemos pasar el test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 20 | onFormSubmit: *function*() { | | 21 | this.secretWord = this.wordInput.val(); | | 22 | *var* formatedSecretWord = this.formatSecretWord(this.secretWord); | | 23 | this.wordOutput.html(formatedSecretWord); | | 24 | this.createButtons(); | | 25 | },  …… | | 32 | tryLetter: *function*(*letter*) { | | 33 | if (this.secretWord.indexOf(letter)!=-1) { | | 34 | this.wordOutput.html(letter); | | 35 | } | | 36 | } | |

* Refactorizamos los tests.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 52 | *function* playWith(*secretWord*) { | | 53 | wordInput.val(secretWord); | | 54 | wordForm.submit(); | | 55 | } | | 56 |  | | 57 | describe("try letters", *function*() { | | 58 | it("should show the letter 'g' when the letter match", *function*() { | | 59 | playWith("g"); | | 60 |  | | 61 | game.tryLetter("g"); | | 62 | wordOutput.html().should.equal('g'); | | 63 | }); | | 64 | it("should show underscore when the letter does not match", *function*() { | | 65 | playWith("g"); | | 66 |  | | 67 | game.tryLetter("e"); | | 68 | wordOutput.html().should.equal('\_'); | | 69 | }); | | 70 | }); | |

* Creamos el siguiente test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 70 | it("should show only the letters that match", *function*() { | | 71 | playWith("ga"); | | 72 |  | | 73 | game.tryLetter("g"); | | 74 | wordOutput.html().should.equal('g\_'); | | 75 | }); | |

* Hacemos pasar el test( Hacerlo por pasos: Primero this.formated(0,0))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 20 | onFormSubmit: *function*() { | | 21 | this.secretWord = this.wordInput.val(); | | 22 | this.formatedSecretWord = this.formatSecretWord(this.secretWord); | | 23 | this.wordOutput.html(this.formatedSecretWord); | | 24 | this.createButtons(); | | 25 | },  …… | | 32 | tryLetter: *function*(*letter*) { | | 33 | if (this.secretWord.indexOf(letter)!=-1) { | | 34 | *var* word=letter + this.formatedSecretWord.substr(1); | | 35 | this.wordOutput.html(word); | | 36 | } | | 37 | } | |

* Creamos el siguiente test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 76 | it("should show the letter that match in different positions", *function*() { | | 77 | playWith("ga"); | | 78 |  | | 79 | game.tryLetter("a"); | | 80 | wordOutput.html().should.equal('\_a'); | | 81 | }); | |

* Hacemos pasar el test (Hacerlo de a poquito: primero this.formated.substr(0,0) y recién el cambio.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 32 | tryLetter: *function*(*letter*) { | | 33 | *var* index = this.secretWord.indexOf(letter); | | 34 | if(index != -1) { | | 35 | *var* word = this.formatedSecretWord.substr(0, index) + | | 36 | letter + | | 37 | this.formatedSecretWord.substr(index + 1); | | 38 | this.wordOutput.html(word); | | 39 | } | | 40 | } | |

* Creamos el siguiente test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 82 | it("it("should show consecutive letters that match of multiple trials", *function*() { | | 83 | playWith("gato"); | | 84 |  | | 85 | game.tryLetter("g"); | | 86 | game.tryLetter("a"); | | 87 | wordOutput.html().should.equal('ga\_\_'); | | 88 | }); | |

* Hacemos pasar el test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 32 | tryLetter: *function*(*letter*) { | | 33 | *var* index = this.secretWord.indexOf(letter); | | 34 | if(index != -1) { | | 35 | this.formatedSecretWord = this.formatedSecretWord.substr(0, index) + | | 36 | letter + | | 37 | this.formatedSecretWord.substr(index + 1); | | 38 | this.wordOutput.html(this.formatedSecretWord); | | 39 | } | | 40 | } | |

* Creamos el siguiente test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 94 | it("should show consecutive letters that match of multiple trials", *function*(){ | | 95 | playWith('gato'); | | 96 |  | | 97 | game.tryLetter('g'); | | 98 | game.tryLetter('a'); | | 99 | wordDisplayed.html().should.equal("ga\_\_"); | | 100 | }); | |

* Hacemos pasar el test (crear el for encima de todo, reemplazar i por index)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 32 | tryLetter: *function*(*letter*) { | | 33 | for(*var* index = 0; index < this.secretWord.length; index++) { | | 34 | if(letter == this.secretWord[index]) { | | 35 | this.formatedSecretWord = this.formatedSecretWord.substr(0, index) + | | 36 | letter + | | 37 | this.formatedSecretWord.substr(index + 1); | | 38 | this.wordOutput.html(this.formatedSecretWord); | | 39 | } | | 40 | } | | 41 | } | |

* Creamos un test para integrar todo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 103 | it("should try the letter when click the letter button", *function*(){ | | 104 | *var* called=false; | | 105 | *game*.tryLetter=*function*() { | | 106 | called=true; | | 107 | }; | | 108 | game.createButtons(); | | 109 |  | | 110 | $('button:contains("a")', buttonList).click(); | | 111 |  | | 112 | called.should.equal(true); | | 113 | }); | |
| |  |  | | --- | --- | | 97 | it("should show the result after try a letter", *function*(){ | | 98 | playWith("a"); | | 99 |  | | 100 | $('button:contains("a")', buttonList).click(); | | 101 |  | | 102 | wordOutput.html().should.equal('a'); | | 103 | }); | |

* Creamos el código que pase el test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 26 | createButtons: *function*() { | | 27 | for(*var* i = 97; i <= 122; i++) { | | 28 | *var* button = $('<button>' + *String*.fromCharCode(i) + '</button>'); | | 29 | button.click($.proxy(this.onButtonClick,this)); | | 30 | this.buttonList.append(button); | | 31 | } | | 32 | }, | | 33 | onButtonClick:*function*(*e*) { | | 34 | *var* letter=$(*e*.target).html(); | | 35 | this.tryLetter(letter); | | 36 | }, | |

* Probamos que el feature test pase.
* Ejecutar grunt nuevamente
  + Corregir el método tryletter(agregar 3 iguales): *if(letter === this.secretWord[index]) {*

# Refactoring

* Renombrar Game a GameController en los clase y en los tests. Ayudarse de ALT+F3
* Crear la clase Game encima de la clase GameController.
* Mover la propiedad this.secretWord a Game
  + Ingresar como parámetro el secretWord

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | *function* Game (*secretWord*) { | | 2 | this.secretWord=secretWord; | | 3 | } | |

* + Instanciar Game dentro de OnFormSubmit y pasarle como parámetro this.wordInput.val()
  + Reemplazar this.secretWord x this.game.secretWord. Ayudarse de ALT+F3
  + Eliminar la línea this.secretWord=this.wordInput.val()
* Mover el método formatSecretWord()
  + Examinar si el método se está utilizando, crear un nuevo setup() e instanciar la clase a la cuál se está moviendo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 3 | describe("Game", *function*(){ | | 4 | *var* game; | | 5 |  | | 6 | beforeEach(*function*(){ | | 7 | game=new Game(''); | | 8 | }); | | 9 |  | | 10 | it("should format the secret word with dashes", *function*() { | | 11 | *var* word = game.formatSecretWord('gato'); | | 12 |  | | 13 | word.should.equal('\_\_\_\_'); | | 14 | }); | | 15 | }); | |

* + Copiar y pegar el método a la nueva clase para hacer pasar los tests. No cortar.
  + Arreglar las referencias
  + Eliminar el método duplicado.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 27 | onFormSubmit: *function*(*e*) { | | 28 | e.preventDefault(); | | 29 | this.game=new Game(this.wordInput.val()); | | 30 | this.formatedSecretWord = this.game.formatSecretWord(this.game.secretWord); | | 31 | this.wordOutput.html(this.formatedSecretWord); | | 32 | this.createButtons(); | | 33 | }, | |

* Mover la propiedad
  + Antes de mover debemos verificar de que otras clases depende la propiedad al momento de instanciarse y de acuerdo a eso decidir como vamos a instanciar esta propiedad en la nueva clase. Para nuestra suerte depende de métodos de la misma clase a la cuál queremos moverlo.
  + Entonces podemos mover toda la línea. Extrayendo un método y realizando la llamada o aprovechando q instanciamos la clase Game en la línea superior, utilizar ese constructor. Utilicemos el constructor.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | *function* Game (*secretWord*) { | | 2 | this.secretWord=secretWord; | | 3 | this.formatedSecretWord = this.formatSecretWord(this.secretWord); | | 4 | } | |

* + Arreglamos las referencias y luego eliminamos la línea.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 28 | onFormSubmit: *function*(*e*) { | | 29 | e.preventDefault(); | | 30 | this.game=new Game(this.wordInput.val()); | | 31 | this.wordOutput.html(this.game.formatedSecretWord); | | 32 | this.createButtons(); | | 33 | }, | |

* Mover el for del método tryletter
  + Crear un método tryLtr
  + Examinar de que variables locales depende y que valores de retorno.
  + Extraer el contenido del tryLetter en el tryLtr y arreglar la referencia.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 45 | tryLetter: *function*(*letter*) { | | 46 | this.tryLtr(letter); | | 47 | this.wordOutput.html(this.game.formatedSecretWord); | | 48 | }, | | 49 | tryLtr:*function*(*letter*) { | | 50 | for(*var* index = 0; index < this.game.secretWord.length; index++) { | | 51 | if(letter === this.game.secretWord[index]) { | | 52 | this.game.formatedSecretWord = this.game.formatedSecretWord.substr(0, index) + | | 53 | letter + | | 54 | this.game.formatedSecretWord.substr(index + 1); | | 55 | } | | 56 | } | | 57 | } | |

* + Antes de moverlo a la otra clase, examinamos dependencias a propiedades de la clase actual. Utilizar el primer This y ALT+f3. Podemos observar que este método principalmente depende de propiedades de otra clase this.game. Por lo tanto es buena idea moverlo a otro lugar. Feature Envy.
  + Mover el método , remplazar los this.game x this.
  + Actualizar la referencia.
  + Renombramos el método a tryLetter.
* Mover a nuevas clases.
  + Crear el archivo gamecontroller.js y mover el contenido, guardarlo.
  + Agregar la referencia en el unit\_runner.html
  + Crear el archivo gamecontroller\_spec.js y mover el contenido, guardarlo.
  + Arreglar el global del header de los specs.
  + Ejecutar Feature Runner
    - Renombrar en el spec de Game a GameController
    - Agregar la referencia en el feature\_runner.html a las nuevas clases
  + Ejecutar Grunt

# 3er Feature Test

* Refactorizamos los Feature Tests para agregar el tryLetter().

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 30 | *function* tryLetter (*letter*) { | | 31 | $('button:contains("'+letter+'")', buttonList).click(); | | 32 | } | | 33 | …… | | 40 | it("should be able to play", *function*() { | | 41 | playWith('cocodrillo'); | | 42 |  | | 43 | tryLetter('o'); | | 44 | tryLetter('d'); | | 45 |  | | 46 | wordOutput.html().should.equal('\_o\_od\_\_\_\_o'); | | 47 | }); | |

* Escribimos el Feature Tests.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 49 | it("should show the number of failed attempts",*function*() { | | 50 | playWith('cocodrillo'); | | 51 |  | | 52 | tryLetter('a'); | | 53 | tryLetter('b'); | | 54 |  | | 55 | failedAttemptsOutput.html().should.equal('2'); | | 56 | }); | |

* Completamos el BeforeEach hasta ver el aserto correcto.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 10 | beforeEach(*function*() { | | 11 | wordInput = $('<input>'); | | 12 | wordForm = $('<form>'); | | 13 | wordOutput = $('<div>'); | | 14 | buttonList=$('<div>'); | | 15 | failedAttemptsOutput=$('<span>'); | | 16 | gameController = new GameController({ | | 17 | wordInput: wordInput, | | 18 | wordForm: wordForm, | | 19 | wordOutput: wordOutput, | | 20 | buttonList: buttonList, | | 21 | failedAttemptsOutput:failedAttemptsOutput | | 22 | }); | | 23 | }); | |

* Vamos a hacer pasar el Feature Tests, pero ahora las clases que manejan el flujo de la aplicación y la lógica de la misma están separadas, entonces comenzamos con la clase Gamecontroller.
* La clase GameController se encargará de mostrar el valor de intentos fallidos.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | 103 | it("should show the failed attemps", *function*(){ | | 104 | gameController.tryLetter("any\_lettetr"); | | 105 |  | | 106 | failedAttemptsOutput.html().should.equal('3'); | | 107 | }); | | |

* Agregamos el failedAttemptsOutput como parámetro.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 10 | beforeEach(*function*() { | | 11 | wordInput = $('<input>'); | | 12 | wordForm = $('<form>'); | | 13 | wordOutput = $('<div>'); | | 14 | buttonList = $('<ditv>'); | | 15 | failedAttemptsOutput = $('<span>'); | | 16 | gameController = new GameController({ | | 17 | wordInput: wordInput, | | 18 | wordForm: wordForm, | | 19 | wordOutput: wordOutput, | | 20 | buttonList: buttonList, | | 21 | failedAttemptsOutput:failedAttemptsOutput | | 22 | }); | | 23 | }); | |

* Si observamos el error nos dice que no hemos inicializado ciertas propiedades de la clase Game. Si revisamos el método TryLetter, veremos que esté hace una llamada a un método de la clase Game y probablemente este método esté necesitando estas propiedades.
* Para implementar el test, necesitamos considerar lo siguiente. A través de este test necesitamos probar la clase GameControlle de manera independiente pero esta necesita a la clase Game para que le proporcione el valor. Para lograr esto necesitamos hacer Mocking o stubbing de las métodos a través de las cuales se comunica con la clase Game.
* Debido a las características propias de JS podemos crear un fake object de manera muy fácil sobreescribiendo la propiedad game.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 103 | it("should show the failed attemps", *function*(){ | | 104 | gameController.game={ | | 105 | tryLetter:*function*() { | | 106 | //do nothing | | 107 | } | | 108 | }; | | 109 |  | | 110 | gameController.tryLetter("any\_letter"); | | 111 |  | | 112 | failedAttemptsOutput.html().should.equal('2'); | | 113 | }); | |

* Pero aún tenemos un problema para terminar de implementar este test, si observamos el aserto este indica que debemostrar 3 errores, pero de donde provienen estos errores o xq 2. En esta oportunidad este valor proviene de la clase Game, entonces en el test necesitamos indicar que el valor de intentos fallidos que retorne la clase Game sea el mismo valor que se está mostrando en la pantalla.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 103 | it("should show the failed attemps", *function*(){ | | 104 | gameController.game={ | | 105 | tryLetter:*function*() { | | 106 | //do nothing | | 107 | }, | | 108 | getFailedAttempts:*function*() { | | 109 | return 2; | | 110 | } | | 111 | }; | | 112 |  | | 113 | gameController.tryLetter("any\_letter"); | | 114 |  | | 115 | failedAttemptsOutput.html().should.equal('2'); | | 116 | }); | |

* Problemas:
  + Variables temporales y más código del que debería ser necesario en el test.
  + Estamos afectando directamente las clases sobre las cuales se están ejecutando los tests, entonces muchas veces necesitamos revertir estos cambios para evitar problemas.
* Agregamos Sinon.Js en las referencias del Unit\_Runner.html
* De esta manera estamos creando un fakeobject de la clase Game y creará un stub para todos los métodos de esta clase que simplemente no hagan nada. Pero si queremos decirle que alguno de estos métodos devuelva algún valor particular también lo podemos hacer.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 103 | it("should show the failed attemps", *function*(){ | | 104 | *var* game=sinon.stub(gameController.game); | | 105 | game.getFailedAttempts.returns(2); | | 106 |  | | 107 | gameController.tryLetter("any\_letter"); | | 108 |  | | 109 | failedAttemptsOutput.html().should.equal('2'); | | 110 | }); | |

* Hacemos pasar el test.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 33 | tryLetter: *function*(*letter*) { | | 34 | this.game.tryLetter(letter); | | 35 | this.wordOutput.html(this.game.formatedSecretWord); | | 36 | this.failedAttemptsOutput.html(this.game.getFailedAttempts()); | | 37 | } | |

* Creamos el primer test en la clase Game.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 16 | describe("Failed Attemps", *function*() { | | 17 | it("should return 1 failed attempt when try 1 incorrect letter", *function*() { | | 18 | game.start('gato'); | | 19 |  | | 20 | game.tryLetter('e'); | | 21 |  | | 22 | game.getFailedAttempts().should.be.equal(1); | | 23 | }); | | 24 | }); | |

* Pasamos el test.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 30 | getFailedAttempts:*function*() { | | 31 | return 1; | | 32 | } | |

* Failed Test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 25 | it("should return 2 failed attempt when try 2 incorrect letters", *function*() { | | 26 | game.start('gato'); | | 27 |  | | 28 | game.tryLetter('e'); | | 29 | game.tryLetter('i'); | | 30 |  | | 31 | game.getFailedAttempts().should.be.equal(2); | | 32 | }); | |

* Pasamos el test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 6 | start:*function*(*secretWord*) { | | 7 | this.secretWord=secretWord; | | 8 | this.formatedSecretWord = this.formatSecretWord(this.secretWord); | | 9 | this.failedAttempts=0; | | 10 | },  …… | | 18 | tryLetter: *function*(*letter*) { | | 19 | for(*var* index = 0; index < this.secretWord.length; index++) { | | 20 | if(letter === this.secretWord[index]) { | | 21 | this.formatedSecretWord = this.formatedSecretWord.substr(0, index) + | | 22 | letter + | | 23 | this.formatedSecretWord.substr(index + 1); | | 24 | } | | 25 | } | | 26 | this.failedAttempts=this.failedAttempts+1; | | 27 | }, | | 28 | getFailedAttempts:*function*() { | | 29 | return this.failedAttempts; | | 30 | } | |

* Failed Test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 34 | it("should return 2 failed attempt when try 2 incorrect and 1 correct letter", *function*() { | | 35 | game.start('gato'); | | 36 |  | | 37 | game.tryLetter('a'); | | 38 | game.tryLetter('e'); | | 39 | game.tryLetter('i'); | | 40 |  | | 41 | game.getFailedAttempts().should.be.equal(2); | | 42 | }); | |

* Pasamos el test.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 18 | tryLetter: *function*(*letter*) { | | 19 | for(*var* index = 0; index < this.secretWord.length; index++) { | | 20 | if(letter === this.secretWord[index]) { | | 21 | this.formatedSecretWord = this.formatedSecretWord.substr(0, index) + | | 22 | letter + | | 23 | this.formatedSecretWord.substr(index + 1); | | 24 | } | | 25 | } | | 26 | if (this.secretWord.indexOf(letter)==-1) { | | 27 | this.failedAttempts=this.failedAttempts+1; | | 28 | } | | 29 | }, | | 30 | getFailedAttempts:*function*() { | | 31 | return this.failedAttempts; | | 32 | } | |

* Refactoring
  + Extraemos el método *updateFormatedSecretWord*
  + Extraemos condicional a método *letterMatch*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | 18 | updateFormatedSecretWord:*function*(*letter*) { | | 19 | for(*var* index = 0; index < this.secretWord.length; index++) { | | 20 | if(letter === this.secretWord[index]) { | | 21 | this.formatedSecretWord = this.formatedSecretWord.substr(0, index) + | | 22 | letter + | | 23 | this.formatedSecretWord.substr(index + 1); | | 24 | } | | 25 | } | | 26 | }, | | 27 | letterMatches:*function*(*letter*) { | | 28 | return this.secretWord.indexOf(letter)!=-1; | | 29 | }, | | 30 | tryLetter: *function*(*letter*) { | | 31 | this.updateFormatedSecretWord(letter); | | 32 | if (!this.letterMatches(letter)) { | | 33 | this.failedAttempts=this.failedAttempts+1; | | 34 | } | | 35 | }, | | |

# Change Feature: Ajax Service

* Problemas
  + Si el servicio falla, también lo harán todos mis tests si que hayas cambiado nada y eso puede resultar confuso. No queremos esos tipos de tests.
* Por defecto mocha falla si se agregan variables globales, en este caso sinon internamente agrega el XMLHttpRequest.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 44 | mocha.globals(['XMLHttpRequest']).run(); | |

* Agregar sinon a los globals de Grunt.

/\*global GameController:false, sinon:false\*/

* Como es una nueva funcionalidad tenemos que comenzar su implementación escribiendo nuevos tests. En esta oportunidad como va a remplazar a una antigua funcionalidad tenemos 2 opciones:
  + Eliminar los tests anteriores y empezar a crear nuevos tests.
  + Empezar a agregar tests de manera paralela a los antiguos, con lo cuál la nueva funcionalidad también va a convivir con la anterior.
* Para decidir cuál es el enfoque adecuado vamos a examinar cuales son los tests que va a ser reemplazados. Observamos que la funcionalidad afectada no son muchos tests.
* No eliminamos completamente el primer tests sino remplazamos su contenido con el nuevo comportamiento que debe tener la clase.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 37 | it("should show the secret word", *function*() { | | 38 | server.respondWith('{"secretword":"gato"}'); | | 39 |  | | 40 | gameController.start(); | | 41 | server.respond(); | | 42 |  | | 43 | wordOutput.html().should.equal('\_\_\_\_'); | | 44 | }); | |

* Escribimos el código que haga pasar este tests. Pero no hace pasar el tests.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 16 | start:*function*() { | | 17 | $.getJSON('/url', *function*(data) { | | 18 | this.game.start(data.secretword); | | 19 | this.wordOutput.html(this.game.formatedSecretWord); | | 20 | }); | | 21 | }, | |

* Podemos observar que efectivamente se está realizando la llamada asíncrona.

Console.log(“azync”);

* Cuando se trata de callback asyn los errores no siempre son mostrados en la consola. Pero podemos ver el navegador.
* El problema es que this dentro de una llamada Ajax no referencia al objeto actual.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 16 | start:*function*() { | | 17 | *var* self=this | | 18 | $.getJSON('/url', *function*(data) { | | 19 | self.game.start(data.secretword); | | 20 | self.wordOutput.html(self.game.formatedSecretWord); | | 21 | }); | | 22 | }, | |

* Ahora el playWith está directamente asociado a la funcionalidad que deseamos reemplazar, podríamos hacerlo paulatinamente. Creando un playWith2 y empezar a reemplazarlo tests por test, o reemplazar directamente el playwith con nuevo contenido.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 28 | *function* playWith(*secretWord*) { | | 29 | server.respondWith('{"secretword":"'+secretWord+'"}'); | | 30 |  | | 31 | gameController.start(); | | 32 | server.respond(); | | 33 | } | |

* Como al reemplazar el contenido de playWith no se producen muchos errores. El problema es que no se crean los botones.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 16 | start:*function*() { | | 17 | *var* self=this; | | 18 | $.getJSON('/url', *function*(data) { | | 19 | self.game.start(data.secretword); | | 20 | self.wordOutput.html(self.game.formatedSecretWord); | | 21 | self.createButtons(); | | 22 | }); | | 23 | }, | |

* Pasan los tests unitarios y feature.
* Ejecutamos la funcionalidad en el navegador y grunt.

# Feature 4

* Feature Tests.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 68 | it("should prevent to play the same letter twice", *function*(){ | | 69 | playWith("cocodrillo"); | | 70 |  | | 71 | tryLetter("a"); | | 72 | tryLetter("a"); | | 73 |  | | 74 | failedAttemptsOutput.html().should.equal('1'); | | 75 | }); | |

* Failed Unit Tests.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 119 | it("should block the button after a try", *function*(){ | | 120 | sinon.stub(gameController.game); | | 121 | gameController.createButtons(); | | 122 | *var* button=$('button:contains("a")', buttonList); | | 123 |  | | 124 | button.click(); | | 125 |  | | 126 | button.is(':visible').should.equal(false); | | 127 | }); | |

* El test pasa directamente sin que hayamos realizado algún cambio, esto se debe a que los elementos html que ingresamos como parámetro al controller son solo variables en memoria y no existen realmente dentro del DOM, y determinados tests no funcionarán correctamente, especialmente aquellos que tengan que ver con visibilidad y dimensiones.
* Agregamos los elementos al DOM, este es un aspecto muy importante que debemos recordar cuando hagamos pruebas JS que interactúen con el DOM
* Podríamos realizar lo siguiente, pero es un proceso bastante engorroso, incluso utilizando jquery para hacer esto seguiría siendo igual de engorroso.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 20 | *var* element=*document*.createElement('div'); | | 21 | element.setAttribute('id','letters'); | | 22 | *document*.body.appendChild(element); | | 23 | buttonList=$('#letters'); | |

* Al testing framerworks nos ofrecen la habilidad de agregar elementos al DOM. En este caso vamos a utilizar js-fixtures.
  + Comentamos el tests.
  + Agregamos la referencia en el unit\_runner.
  + Agregamos una función que cargue los tests.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 10 | beforeEach(*function*() { | | 11 | *var* fixture=loadFixture(); | | 12 | …… | | 23 | }); | | 28 |  | | 29 | *function* loadFixture(){ | | 30 | fixtures.path='fixtures'; | | 31 | fixtures.load("game.html"); | | 32 | return $('#' + fixtures.containerId).contents().find('body'); | | 33 | } | |

* + Refactorizamos el beforeEach.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 10 | beforeEach(*function*() { | | 11 | *var* fixture=loadFixture(); | | 12 |  | | 13 | wordOutput = $('#wordDisplayed',fixture); | | 14 | buttonList = $('#letters',fixture); | | 15 | failedAttemptsOutput = $('#failedAttempts',fixture); | | 16 | gameController = new GameController({ | | 17 | wordOutput: wordOutput, | | 18 | buttonList: buttonList, | | 19 | failedAttemptsOutput:failedAttemptsOutput | | 20 | }); | | 21 |  | | 22 | server=sinon.fakeServer.create(); | | 23 | }); | |

* + Agregamos un afterEach.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 25 | afterEach(*function*(){ | | 26 | fixtures.cleanUp(); | | 27 | }); | |

* Luego de realizar los cambios el tests ya falla.
* Implementamos el código que pase el tests.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 30 | onButtonClick:*function*(*e*) { | | 31 | *var* letter=$(e.target).html(); | | 32 | this.tryLetter(letter); | | 33 | $(e.target).remove(); | | 34 | }, | |

* Refactorizamos el Feature\_Spec también para que haga uso del fixture (OPCIONAL)
* Ejecutamos el build y probamos directamente en la pantalla.

# Coverage

* Instrumentar nuestro código con coverJs
  + npm install -g coverjs
  + coverjs.cmd -o ./output/instrument ./public/app/ -r
* Copiamos el unit\_runner a coverage.
* Agregar las referencias de JsCovReporter al Html (JS y CSS)

<link rel="stylesheet" href="lib/JSCovReporter/reporter.css">

<script type="text/javascript" src="lib/JSCovReporter/JSCovReporter.js"></script>

* Agregamos las referencias al código instrumentado.
* Agregamos el los tags HTML para que imprima el reporte.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 13 | <div id="coverage"></div> | | 14 | <div id="menu"></div> | |

* Configuramos Mocha para que ejecute el reporte de Coverage.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 54 | <script type="text/javascript"> | | 55 | *function* coverageReport() { | | 56 | *var* reporter = new JSCovReporter(); | | 57 | reporter.initialize({ | | 58 | coverObject: *window*.\_\_$coverObject | | 59 | }) | | 60 | } | | 61 | mocha.globals(['XMLHttpRequest','jQuery\*']).run(coverageReport); | | 62 | </script> | |

# Deployment

* Instalar grunt

Npm i –g grunt

* Tasks adicionales para Grunt

Npm i grunt-contrib

Npm i grunt-bump

Npm i grunt-exec

* Instalar jitsu

Npm i –g jitsu

* Deployar

Jitsu deploy

# Referencias

* Stubbing SinonJS:

<http://sinonjs.org/>

<http://msdn.microsoft.com/en-us/magazine/gg649850.aspx>

<http://joseoncode.com/2012/01/18/sinonjs-mocks-for-javascript/>

* Unit Testing Fixtures

<https://github.com/badunk/js-fixtures/>

<https://github.com/velesin/jasmine-jquery/>

* Coverage

<https://github.com/TwoApart/JSCovReporter>

<https://github.com/lawrencec/JSCovReporter> (sin dependencia a backbone)

<https://github.com/visionmedia/node-jscoverage>

<https://github.com/arian/CoverJS>

* Build Tool

<http://gruntjs.com/>

<https://github.com/mde/jake>

<http://brunch.io/>

* JSHint

<https://github.com/derekprior/jshint-junit-xml-formatter>

