

**Trabajo Practico Validación y Verificación de Programas**

**Alumno:** Nicolas Guaquin.

**Profesor:** Silvina Gil Romero

**Fecha de entrega:** 06/11/2020.

**Carrera:** Técnico Superior en Desarrollo de Software.

**Introducción a Pruebas Unitarias:**

Es una forma de comprobar el correcto funcionamiento de una unidad de código, Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado.

El objetivo es aislar cada parte del programa y mostrar que las partes individuales son correctas.

**Qunit**

**QUnit**, es un marco genérico para probar cualquier código JavaScript, para pruebas de Unidad, por eso es que se dice que es robusto y tiene un gran alcance.

Desarrollado originalmente para probar/testear librerias [jQuery](https://en.wikipedia.org/wiki/JQuery) , [jQuery UI](https://en.wikipedia.org/wiki/JQuery_UI) y [jQuery Mobile](https://en.wikipedia.org/wiki/JQuery_Mobile) Admite entornos del [lado del cliente](https://en.wikipedia.org/wiki/Client-side) en navegadores web y del lado del servidor (por ejemplo, [Node.js](https://en.wikipedia.org/wiki/Node.js) ).

QUnit fue desarrollado originalmente por [John Resig](https://en.wikipedia.org/wiki/John_Resig) como parte de jQuery. En 2008 se extrajo del código fuente de la prueba unitaria jQuery para formar su propio proyecto y se conoció como "QUnit". Esto permitió que otros comenzaran a usarlo para escribir sus [pruebas unitarias](https://en.wikipedia.org/wiki/Unit_tests) .

**Como incluir QUnit**

Para empezar a utilizarlo, se necesita incluir en nuestros ficheros, dos archivos:

Un Script con la librería y una hoja de estilos (opcional) que permite una correcta virualizacion de los resultados.

**Qunit y sus atributos**

**Afirmaciones o Assertions'**

**Agrupacion de pruebas Pruebas o Grouping Test'**

* **A*firmaciones:*** Son bloques que componen nuestros test**, que evaluan el resultado de una determinada porción de un código**. Si el resultado obtenido es correcto (true), el test se supera; si obtenemos un valor falso (false), el test no se supera.

**Existen 3 tipos de afirmaciones**

* OK (verdadero,mensaje); Afirma que el valor proporcionado se [convierte](https://en.wikipedia.org/wiki/Type_conversion) en [booleano verdadero](https://en.wikipedia.org/wiki/Boolean_data_type) .
* equal (valor actual, valor esperado,mensaje);:  Compara dos valores, utilizando el [operador de doble igualdad](https://en.wikipedia.org/wiki/Double-equal_operator) .
* deepEqual (valor actual, valor esperado, mensaje); Compara dos valores según su contenido, no solo su identidad.

**Estructura afirmación OK**

ok( state, message )

* **state**(*Boolean*) Expresión booleana o de cualquier otro tipo que será evaluada.
* **message**[opcional] (*String*) Mensaje de salida que acompaña el resultado de la afirmación.

**Ejemplo:**

test("funciones basicas",function(){  
var one = 2;  
var two = 2;  
ok( one == two, "Excelente");  
})

(verifica si ambas variables tienen el mismo valor.)

**Agrupación de una o mas Pruebas.**

La agrupación de pruebas es una manera de segmentar el conjunto de tests que hemos realizado durante el desarrollo del proyecto,

Con la función **module()** podemos agrupar las pruebas unitarias por medio de una descripción o característica que tengan las mismas.

Estructura:

module( name, [lifecycle] )

Donde:

* **name**(*String*) es el nombre del módulo.
* **lifecycle**[opcional] (*Options*) : Permite establecer callbacks que se ejecutarán tanto antes como después de cada test que compone este módulo

Ejemplo:

module( "Test Module A" );

test( 'Testing checkNumber()', **function**() {

ok( checkNumber( 0 ) );

ok( checkNumber( 2 ) );

ok( checkNumber( -4 ) );

ok( checkNumber( 1 ) );

ok( checkNumber( 'asdf' ) );

ok( !checkNumber( 'asdf' ));

} );