Entrega 3 - Objetos y closures

Versión: 12 de Febrero de 2019

Objetivo

Practicar con la definición y la instanciación de módulos creados con closures.

Descripción de la práctica

El fichero **mod3-stock_obj_closure.js** incluye el esqueleto dado a continuación, donde falta el código que implementa los métodos **get_p(code)**, **del_p(code)** y **addJSON(json_prods)**. Este programa debe implementar un control de stocks de productos.

Incluir el código en el fichero **mod3-stock_obj_closure.js** incluido en el proyecto de prueba: https://github.com/practicas-ging/mooc_node-mod3_stock_obj_closure (ver sección "Realización y prueba de la práctica").

Añadir el código de los métodos incompletos (get_p(code), del_p(code) y addJSON(json_prods)) para que al utilizar este gestor de stocks con el código de la última parte del programa, se obtenga como resultado de su ejecución que se muestra en la captura que viene después del código.

```
========== mod3-stock_obj_closure.js: Comienzo del esqueleto del código  ============

function stock (title) {
    let    _title = title;  //  Title of stock manager
    const  _stock = {};     //  prods: { <code>: {c: code, desc: <description>,  n: <number>}

         // Returns access object to internal state (variables), uses ES6 object method syntax
    return {

        title  ()  {              // Returns title of stock manager
            return _title;
        },

        new_p (code, desc) {      // Adds product to stock if not yet included
            if (!_stock[code]) {  // if already included returns null, else product object
                _stock[code] = {code, desc, n:0};
                return _stock[code];
            };
            return null;
        },

        add (code, n) {           // if product exists add n and return product, else return null
            if (_stock[code]) {
                _stock[code].n += n;
                return _stock[code];
            }
            return null;
        },

       rem (code, n) { // if n prods in stock subtract n and return product object or return null
            if ( _stock[code] && _stock[code].n >= n ) {
                _stock[code].n -= n;
                return _stock[code];
            }
            return null;
        },
```

```
        number () {             // return number of prods (length of array of prod objects)
            return Object.keys(_stock).length;
        },


       get_p (code) {           // return product obj if exists or null if it doesn't
            // ....... add code here
        },

       del_p (code) {           // if code exists eliminate and return it, or return null
            // ....... add code here

            // .......
        },

       addJSON (json_prods) {   // Add n to prod if code exists, or create new prod
            // ....... add code here


            // .......
        },

       getJSON () {             // Returns content of _stock serialised as a JSON string
            return JSON.stringify( _stock );
        },

        reset () {              // Remove all products from _stock
            _stock = {};
        }
    }
}


let my_shop = stock ("My shop");

console.log();
my_shop.new_p('a1', 'fork');
my_shop.add('a1', 3);
my_shop.new_p('a4', 'spoon');
my_shop.add('a4', 7);
console.log("-> my_shop.new_p('a1', 'fork')");
console.log("-> my_shop.add('a1', 3)");
console.log("-> my_shop.new_p('a4', 'spoon')");
console.log("-> my_shop.add('a4', 7)");

console.log();
console.log("There are " + my_shop.number() + " prods");

console.log();
console.log("_stock= " + my_shop.getJSON());

console.log();
console.log();

my_shop.addJSON('{ "a1":{"n":2}, "a2":{"code":"a2", "desc":"knife", "n": 3}}');
console.log(`-> my_shop.addJSON('{ "a1":{"n":2}, "a2":{"code":"a2", "desc":"knife", "n": 3}}'`);

console.log();
console.log("_stock= " + my_shop.getJSON());

console.log();
console.log();

my_shop.add('a1', 4);
console.log("-> my_shop.add('a1', 4)");
```

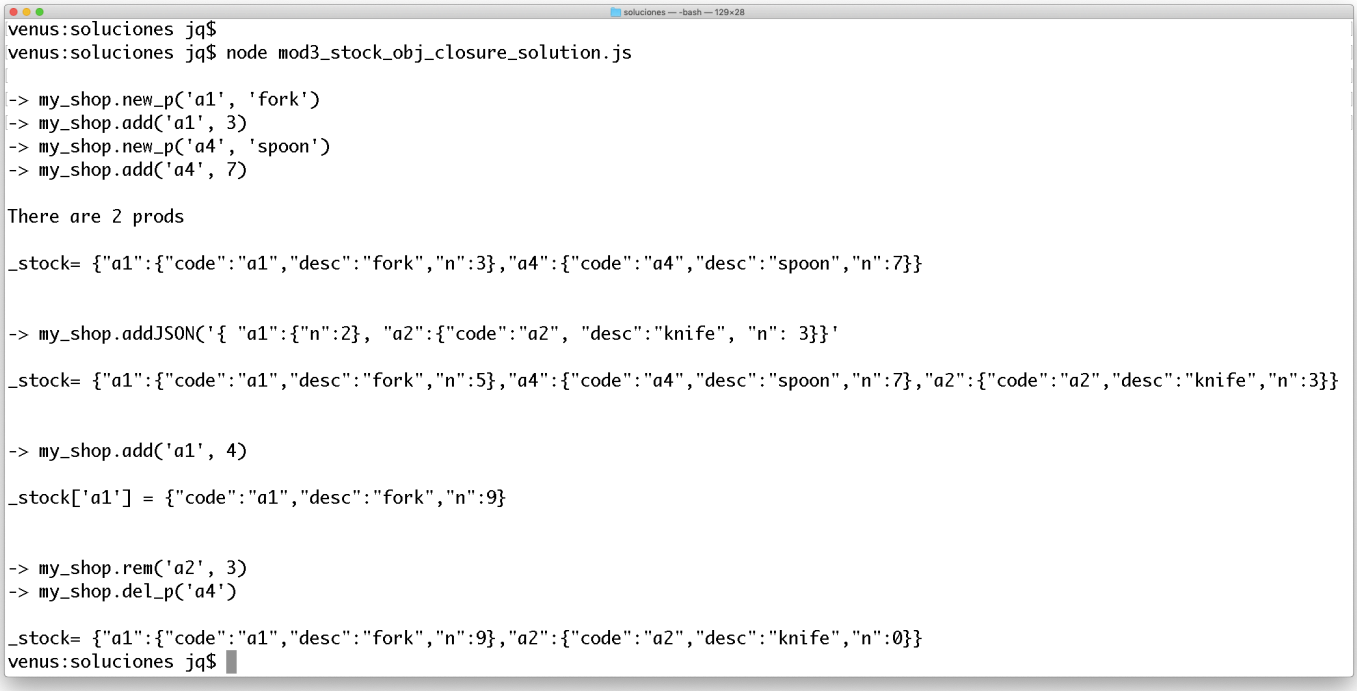
```
  console.log();

  console.log("_stock['a1'] = " + JSON.stringify(my_shop.get_p('a1')));
  console.log();
  console.log();

  my_shop.rem('a2', 3);
  my_shop.del_p('a4');
  console.log("-> my_shop.rem('a2', 3)");
  console.log("-> my_shop.del_p('a4')");

  console.log();
  console.log("_stock= " + my_shop.getJSON());

  ============ Final del código ============
```



Recomendación para implementar get_p(code): si existe, devolver el objeto guardado en la propiedad de nombre “code”, sino devolver null.

Recomendación para implementar del_p(code): si existe el código, borrar del stock y devolver el producto borrado, sino devolver null.

Recomendación para implementar addJSON(json_prods): transformar “json_prods” en un objeto JavaScript equivalente. Iterar en todos los nuevos productos, sumando a los productos que ya están en "_stock" la cantidad de nuevos productos incluidos en la nueva remesa (json_prods) y añadir a "_stock" todos los nuevos productos de “json_prods" que no existen todavía en "_stock".

Posible opción: El gestor de stocks definido como cierre puede transformarse en una clase de nombre Stock con sintaxis ES6. En este caso se debe utilizar el mismo código de uso del cierre, salvo la primera instrucción que deberá ser: let my_shop = new Stock ("My shop”);

Realización y prueba de la práctica

Para comprobar que la práctica ha sido realizada correctamente hay que utilizar el validador de este repositorio

https://github.com/practicas-ging/mooc_node-mod3_stock_obj_closure

Recuerde que para utilizar el validador se debe tener node.js (y npm) (https://nodejs.org/es/) y Git instalados. El proyecto se descarga, instala y ejecuta en el ordenador local con estos comandos:

```
$                                ## El proyecto debe clonarse en el ordenador local
$  git clone https://github.com/practicas-ging/mooc_node-mod3-stock_obj_closure
$
$  cd mooc_node-mod3-stock_obj_closure       ## Entrar en el directorio de trabajo
$
$ npm install                                ## Instala el programa de test
$
$                             ##  ->  Incluir la solución en el esqueleto clonado
$
$ npm run checks                       ]      ## Pasa los tests al fichero solicitado
.....................................    ## en el directorio de trabajo
.....................................
... (resultado de los tests)
$
```

Una vez descargado el proyecto, esta entrega se debe realizar de la siguiente forma. Entrar en el directorio raiz **mod3-stock_obj_closure**. El fichero **mod3-stock_obj_closure.js**, esta incluido (incompleto) en este directorio. Este debe completarse con el editor o sustituirse por otro del mismo nombre que contenga la solución. A continuación deben pasarse los tests para ver si la solución es correcta.

Los tests pueden pasarse las veces que sea necesario. Si se pasan nada más descargar el proyecto, indicarán que no se ha realizado todavía nada de esta practica. También pueden utilizarse para probar el programa de otro compañero sustituyendo los ficheros que se desee probar. El programa de test incluye además un comando para generar el fichero ZIP

```
$
$ npm run zip         ##  Comprime los ficheros del directorio en un fichero .zip
$
```

Este genera el fichero **mooc_node-mod3-stock_obj_closure_entregable.zip** con el directorio de la practica comprimido. Este fichero ZIP debe subirse a la plataforma para su evaluación.

Instrucciones para la Entrega y Evaluación.

Se debe entregar el fichero **mooc_node-mod3-stock_obj_closure_entregable.zip** con los ficheros comprimidos de la entrega.

El evaluador debe descargar el fichero entregado y comprobar que funciona correctamente. El fichero descargado es un paquete npm, que puede instalarse con todas sus dependencias con npm. Una vez instalado puede ejecutarse o pueden pasarse los test.

RUBRICA. Se puntuará el ejercicio a corregir sumando el % indicado a la nota total si la parte indicada es correcta:
- 25%: si el método get_p (code) está bien implementado
- 35%: si el método del_p (code) está bien implementado
- 40%: si el método addJSON (json_prods) está bien implementado

Si pasa todos los tests se deberá dar la máxima puntuación.

El objetivo de este curso es sacar el máximo provecho al trabajo dedicado y para ello lo mejor es utilizar las evaluaciones para ayudar al evaluado, especialmente a los principiantes. Al evaluar se

debe dar comentarios sobre la corrección del código, su claridad, legibilidad, estructuración y documentación, siempre que puedan ayudar al evaluado.

¡Cuidado! Una vez enviadas, tanto la entrega, como la evaluación, no se pueden cambiar. Esperar a tener completa y revisada, tanto la entrega, como la evaluación antes de enviarlas.