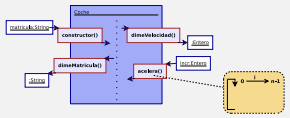
Grado Tecnologías Interactivas

Programación 2



Práctica 3

UNIVERSIDAD POLITECNICA DE VALENCIA

Escola Politècnica Superior de Gandia

DSIC

Departament de Sistemes Informàtics i Computació

Práctica 3

¡ Atención !

▷ Se recuerda que las prácticas deben prepararse antes de acudir al aula informática, anotando en el enunciado las dudas que se tengan.

▷ Los diseños y algortimos que se piden en esta práctica deben escribirse en la libreta de apuntes para poder ser revisados.

▷ Utiliza *git* en cada ejercicio haciendo *commit* cada vez que consigas un "hito".

▷ La realización de las prácticas es un trabajo individual y original. En caso de plagio se excluirá al alumno de la asignatura. Por tanto, es preferible presentar el trabajo realizado por uno mismo aunque éste tenga errores.

1 Objetos y JSON

1.1 Objetos

Aunque lo habitual en otros lenguajes de programación, como C++ o Java, es escribir una clase antes de poder crear objetos de ella; en JavaScript es posible crear un objeto directamente. Por ejemplo:

```
var unCoche = {
 color : "rojo",
 precio : 1234.56
}

console.log( unCoche )
console.log( unCoche.color )
console.log( unCoche.precio )
```

Además, en las propiedades del objeto podemos poner perfectamente funciones:

```
var obj = {
 valor : 1234,
 metodo : function( a ) {
      return this.valor * a
 },
 incrementar : function() {
      this.valor++
 }
} // obj
```

```
obj.incrementar()
var r = obj.metodo( 2 )
console.log( r )
```

Ejercicios

1. Prueba los anteriores ejemplos.
2. Diseña, implementa y prueba un objeto pila (consultar Pila) con operaciones apilar(), desapilar() y cima().

1.2 JSON

Una necesidad habitual de los programas es guardar y recuperar información, así como transmitirla a otros.

Como sabemos, la información manipulada por nuestros programas está almacenada en variables: valores simples, arrays, u objetos. Para poder exportar, importar o enviar los datos guardados en variables necesitamos convertir dicha información a un formato adecuado. Por comodidad, la base de muchos formatos para almacenamiento o transmisión de información es texto (ASCII). Pero como es el caso que la información suele tener estructura (listas u objetos) el formato debe ser capaz no sólo de representar los datos sino también su estructura.

En JavaScript el formato nativo para representar información es JSON: *JavaScript Object Notation*, que no casualmente se asemeja a la sintaxis con la que se crean los objetos y que hemos estudiado en la sección anterior.

Así pues, si creamos el siguente objeto

```
var unaPersona = {
 dni : "20123567R",
 nombre : "Juan",
 apellidos : "García Pérez",
```

```
 edad : 19,
 telefonos : [696234567, 676456123]
}
```

Su representación en JSON es el siguiente texto:

```
'{"dni":"20123567R","nombre":"Juan","apellidos":"García Pérez","edad":19,"telefonos":[696234567,676456123]}'
```

Convertir un objeto a texto JSON es sencillo:

```
var texto = JSON.stringify( unaPersona )
```

E igual de sencillo es convertir un texto en formato JSON a un objeto de JavaScript:

```
var juan = JSON.parse( texto )
```

Ejercicios

1. Prueba los anteriores ejemplos.
2. Diseña e implementa una estructura con objetos de JavaScript para representar tu horario de clases. Utiliza como base el siguiente ejemplo, pero añadiendo la hora de inicio de cada asignatura.

   ```
   var horario = {
    lunes : ["física", "matemáticas", "inglés"],
    martes : ["programación", "matemáticas"],
    miercoles : ["ingles", "física"],
    jueves : ["física", "matemáticas", "inglés"],
    viernes : ["programación", "matemáticas"],
   }
   ```

   Pasa dicha estructura a JSON y escríbela en pantalla.
3. Diseña e implementa una función que reciba un objeto como el del anterior ejercicio, y un nombre de asignatura; y devuelva una lista con los días y las horas en que se imparte dicha asignatura.

2 Ficheros de Texto

Los ficheros de texto sirven para guardar de forma persistente información, en este caso utilizando la tabla ASCII o algun derivado como "utf8".

En JavaScript, éstas son las funciones de biblioteca para

- Crear un fichero de texto

```
var fs = require( "fs" )
fs.writeFile( "hola.txt", "hola mundo", function( err ) {
 if( err ) {
   console.log( "hubo un problema al escribir en hola.txt" )
 }
})
```

- Leer de un fichero de texto

```
var fs = require( "fs" )
fs.readFile( "hola.txt", "utf8", function( err, contenido ) {
 if( err ) {
   console.log( "hubo un problema al leer de hola.txt" )
   return
 }
 console.log( contenido )
})
```

Ejercicios

1. Escribe un programa que escriba tu nombre en un fichero de texto llamado "nombre.txt".
2. Escribe un programa que lea el contenido del fichero "nombre.txt".

3 Caso práctico

Vamos a escribir un programa que simule obtener medidas de temperatura y las guarde en un fichero de texto; y otro que abra el fichero con las temperaturas guardadas y calcule la media de las temperaturas y en qué instante se produjo la máxima y la mínima.

3.1 Mediciones

Estudia con atención el siguiente código que deberás completar.

Fijate que la función `tomarMediciones()` es *recursiva*: se llama a sí misma para simular un bucle. ¿Qué es lo que se utiliza como contador de dicho bucle simulado? ¿Qué es lo que evita que la función se llame a sí misma indefinidamente?

```
// -------------------------
// medirTemperatura() -> JSON{ hora: N, temperatura: R }
//
// Realiza una medida de temperatura y
// devuelve el valor junto con la hora
// -------------------------
function medirTemperatura() {
   //
   // completar: devolver un objeto con dos campos:
   // hora, con la hora actual; y
   // temperatura, un valor aleatorio entre 15 y 20
   //
} // ()
```

Como contador se utiliza cuantas-1. El if con el return dentro

```
// ------------------------------
// cuantas:N -> tomarMediciones() -> Lista<JSON{hora:N, temperatura:R}>
//
// Toma la cantidad de mediciones indicadas llamando
// cada segundo a medirTemperatura()
// ------------------------------
function tomarMediciones( cuantas, mediciones, callback ) {

 if( cuantas == 0 ) {
  callback( mediciones )
  return
 }

 mediciones.push( medirTemperatura() )

 setTimeout( function() {
  tomarMediciones( cuantas-1, mediciones, callback )
 }, 1000 )
} // ()

// -------------------------
// main()
// -------------------------
var medidas = []
//
// completar: llamar a tomarMediciones() para que nos devuelva
// 7 medidas de temperatura y guardar lo que nos devuelve
// en el ficheor "datos.txt" (habiendo convertido los datos
// a JSON previamente)
//
```

3.2 Cálculos

Diseña e implementa un programa que a partir de la información guardada en el fichero "datos.txt" calcule la media de las temperaturas guardadas y cuál es la máxima y la mínima temperatura registrada y las horas en que se produjeron.

Este programa debe tener una función distinta para realizar cada cálculo: calcular la media, encontrar la temperatura máxima y su hora; y encontrar la temperatura mínima y su hora.

13 febrero 2019