Desarrollo Web Entorno Cliente
2º D.A.W.

04. OBJETOS DEFINIDOS POR EL USUARIO

1. Las funciones

- Una función se declara:

```
function  nombreFuncion(parámetro1[=valor],
    parámetro2[=valor],...)
{ instrucción1; instrucción2; ...[arguments.length] [return
    valor;]}
```

- Las funciones se invocan por su nombre seguido de la lista de parámetros concordantes entre paréntesis:

```
nombreFuncion(parámetro1, parámetro2,...)
```

arguments.length: es un array que contiene los argumentos recibidos

- Los argumentos tipo objeto se pasan por **referencia**
- El resto de argumentos se pasan por **valor**
- Las funciones se pueden declarar en cualquier parte de un documento HTML.
- Para poder llamar a una función debe estar declarada en el mismo bloque script o en un script previo.
- Un script se carga por completo en memoria antes de ejecutarse.

1. Las funciones

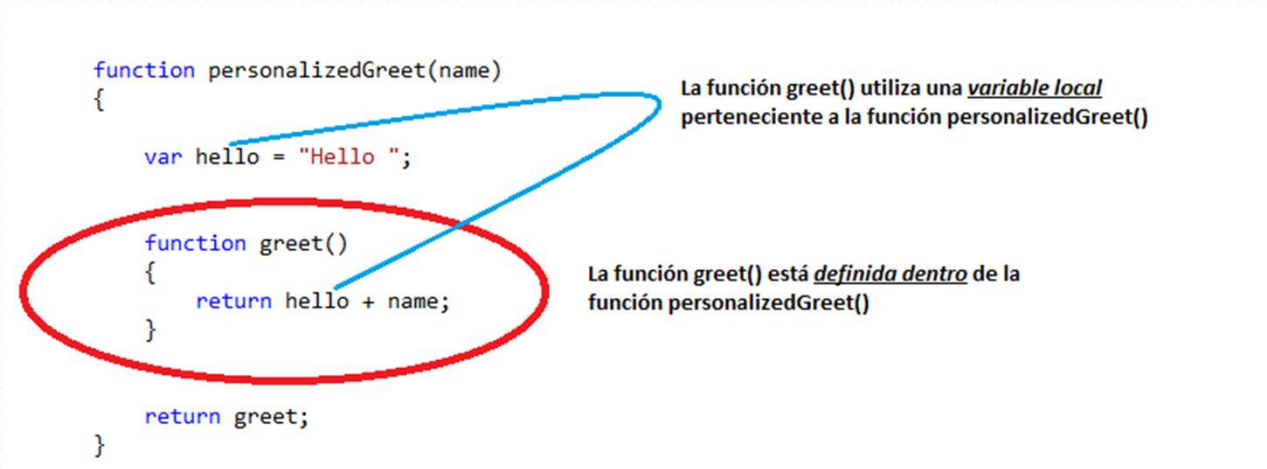
- Si una función devuelve algún valor (return x) se puede usar ésta como asignación.

```
Function miFuncion( ) { ...  return  x }
variable = miFuncion( )
```

- El valor devuelto por la función puede ser de cualquier tipo (cadena, número, boolean, objeto).
- Una función puede utilizar varias sentencias return si hubiese alternativas.
- Las variables declaradas dentro de una función son locales y no son visibles fuera.
- En caso de que una variable global coincida en nombre con una local prevalece la local para la función.

1. Las funciones

- Dentro de una función se puede declarar otra función:



Cuando una variable local se almacena en memoria, si ésta cambia a lo largo de la función externa, en memoria se almacenará el último valor asignado al salir de la función externa.
Cada función interna guarda una referencia a la posición de memoria donde se almacena la variable, pudiendo acceder a su valor.

1. Las funciones

Una función autoejecutable es aquella que se ejecuta al tiempo que se declara.
Para ello se debe encerrar entre paréntesis y se añaden paréntesis al final de la declaración:

```
(function() { alert('Hola Don Pepito'); } )  (  );
```

```
  var y = 3; var x = y;

  (function(){
        //Importante no olvidar punto y coma en la línea anterior!
        } ());
```

Los paréntesis de ejecución ( ) pueden ir dentro o fuera de los paréntesis de la función:

```
(function() { alert('Hola Don Pepito'); } (  ) ) ;
```

Funciones autoejecutables

- en JavaScript se dice que **todo son objetos de una forma bastante literal**, incluso las propias funciones son consideradas objetos que se pueden almacenar en variables.

```
var objeto = {},
var miFuncion = function () {};
```

- Dos ejemplos de **función anónima** –sólo lleva la declaración function- y además **autoejecutable** por los paréntesis que la encierran y se añaden al final.

```
(function () {
    console.log('Esta función no tiene nombre y\
   se ejecutara inmediatamente');
}( ));
```

```
(function (uno, dos, tres) {
        console.log(uno);
        console.log(dos);
        console.log(tres);
}(1, 2, 3));
```

Ejemplo de función anónima autoejecutable

```
alert(function () {
   var s = "" ;

            // Crear encabezado de la tabla.
            s += "hex dec Bin \ n" ;

            for (x = 0; x <16; x ++)   {
               s += "" ;

               // Convertir a hexadecimal.
               s += x.toString (16);
               s += "" ;
               if (x <10) s += "" ;

               // Convertir a decimal.
               s += x.toString (10);
               s += "" ;

               // Convertir a binario.
               s += x.toString (2);
               s += "\ n" ;
            }

            return(s);
}())
```

Ponemos delante un alert para ver en pantalla la ejecución.

2. Arrays

1. Declaración

- **Un array o lista** es una colección de datos ordenada por un índice.
- Para acceder a cada elemento de la colección se usa un mismo nombre pero un subíndice diferente que empieza por 0.

- En JavaScript cada array se declara como una instancia de la clase Array.

```
var  miArray = new Array()   //array con 0 elementos
```

- Array con un nº determinado de elementos (undefined hasta que se asignen sus valores)

```
var miArray = new Array(3)  //array con 3 elementos (0,1,2)
```

- Se puede crear con valores ya asignados:

```
var miArray = new Array(2,4,6,8)  //array con 4 valores dados
var miArray = new Array(21,'cadena',true) //tipos diferentes
var miArray = [2,4,6,8]                   //los corchetes equivalen a new Array
var miArray = [21,'cadena',true]          // "
```

2. Arrays

- Asignar valores a cada elemento:

```
miArray[0] = 290
miArray[1] = 97
miArray[3] = 127
```

- Se pueden asignar desordenadamente, con saltos y superando el índice de la declaración.
  - Si se dejan huecos quedan con el valor 'undefined'
  - Si se usa un índice superior al de la declaración hacemos que aumente la longitud del array.
- Se puede crear un array declarándolo con asignación de otro array

```
Var miArray = [1,3,5]
array2 = miArray
```

- La longitud de un array es el número de elementos que contiene:

```
miArray.length
```

- Recorrer todos los elementos del array:

```
for (i=0;i< miArray.length;i++){
document.write("Posición " + i + " del array: " + miArray[i])
document.write("<br>")
}
```

2. Arrays


```
var tablero = new Array(3);

tablero [0] = new Array(0,1,2); //elem. 0 tiene un array de 3 elementos
tablero [1] = new Array(3,4,5); //elem. 1 tiene un array de 3 elementos
tablero [2] = new Array(6,7,8,9); //elem. 2 tiene un array de 4 elementos
alert(tablero); //mostrará  0,1,2,3,4,5,6,7,8,9
alert(tablero [2][3]); //mostrará el 9
 alert(tablero.length); //el array tablero tiene 3 elementos
 alert(tablero [2].length); //el elem. 2 tiene  un array de 4 elementos
```

2. Arrays

- **length**, devuelve el número de elementos de un array

```
var vocales = ["a", "e", "i", "o", "u"];
var numeroVocales = vocales.length; // numeroVocales = 5
```

- **concat()**, se emplea para concatenar los elementos de varios arrays

```
var array1 = [1, 2, 3];
array2 = array1.concat(4, 5, 6);   // array2 = [1, 2, 3, 4, 5, 6]
array3 = array2.concat(array1);   // array3 = [1, 2, 3, 4, 5, 6, 1, 2, 3]
```

- **join**(separador), devuelve una cadena resultado de unir todos los elementos de un array. Como parámetro lleva un carácter separador que puede ser vacío. Es la función inversa de string.split().

```
var array = ["hola", "mundo"];
var mensaje = array.join(""); // mensaje = "holamundo"
mensaje = array.join("  ");     // mensaje = "hola  mundo"
```

2. Arrays

- **pop()**, elimina el último elemento del array y lo devuelve. El array original se modifica y su longitud disminuye en 1 elemento.

```
var array = [1, 2, 3];
var ultimo = array.pop();
// ahora array = [1, 2],    ultimo = 3
```

- **push()**, añade un elemento al final del array. El array original se modifica y aumenta su longitud en 1 elemento. (También es posible añadir más de un elemento a la vez)

```
 var array = [1, 2, 3];
 array.push(4);
 // ahora array = [1, 2, 3, 4]
```

- **shift()**, elimina el primer elemento del array y lo devuelve. El array original se ve modificado y su longitud disminuida en 1 elemento.

```
var array = [1, 2, 3];
var primero = array.shift();
// ahora array = [2, 3], primero = 1
```

- **unshift()**, añade un elemento al principio del array. El array original se modifica y aumenta su longitud en 1 elemento. (Se puede añadir más de un elemento a la vez)

```
var array = [1, 2, 3];
array.unshift(0);
  // ahora array = [0, 1, 2, 3]
```

2. Arrays

- **reverse(),** modifica un array colocando sus elementos en el orden inverso a su posición original

```
var array = [1, 2, 3];
array.reverse();
// ahora array = [3, 2, 1]
```

- **sort(),** modifica un array ordenando por código ASCII los elementos

```
var colores = new Array(true,'rojo','verde','Rojo',5);
colores.sort();
// devuelve 5, Rojo, rojo, true, verde
```

- **indexOf(elemento),** Devuelve el índice que ocupa el elemento indicado

```
var colores = ['rojo','verde','azul'];
alert(colores.indexOf('verde');
// se muestra el 1,  posición del verde
```

- **LastIndexOf(elemento),** Devuelve el índice que ocupa el elemento indicado

```
var colores = ['rojo','verde','azul','rojo'];
alert(colores.LastindexOf('rojo');
// se muestra el 3,  última posición del 'rojo'
```

2. Arrays

- **slice**(*inicio*[,*fin*]) Devuelve un array con los elementos comprendidos entre el índice *inicio* incluido y el índice *fin* excluido. Si se omite *fin* utiliza todos los elementos a partir de inicio. El array no se modifica.

```
var colores = new Array ('rojo','verde','naranja');
alert(colores.slice(1,2));
// Muestra 'verde'. Elementos entre [1](incluido) y [2] (excluido)
```

- **splice**(*inicio*,*NumEliminar*,[Ins*elemento1*, Ins*elemento2*, ...]) Sirve para **insertar o sustituir** elementos a partir de la posición *inicio* de un array. Si Num*Eliminar* es distinto de 0 se elimina ese número de elementos, y se inserta en su lugar *elemento1*, *elemento2*, ... Si *eliminar* es 0, se insertan *elemento1*, *elemento2*, ... en la posición *inicio* desplazando los elementos existentes hacia la derecha. Si se usa para eliminar devuelve un array con la lista de elementos eliminados; si se usa para insertar no devuelve nada.

```
var colores = new Array('rojo','verde','azul');
colores.splice(1,2,'naranja','rosa','violeta'); //después de 'rojo' elimina 'verde' y 'azul' /añade
   n,r,v
alert(colores);
colores.splice(2,0,'amarillo');
```

2. Arrays

```
miArray = [“a” ,”b”, ”c”]

miArray.forEach(function (elemento, index) {
    console.log(elemento);   //muestra “a", “b", “c"
    console.log(index);          //muestra "0", "1", "2"
});
```

2. Arrays

- Ejercicios planteados:
  1. Array con los nombres de aProvincias (5).
     - Ordenar el array de nombres alfabéticamente, luego inversamente.
     - Localizar la posición de una provincia dando un indicio de su nombre.
  2. Array aNotas de 2 dimensiones con las 3 notas de alumno.
     - Asignar las notas con números aleatorios, obtener la media de cada alumno y agregar dicha nota al array de cada alumno.
     - Insertar delante de cada alumno la fecha de nacimiento

- Ejercicios resueltos:
  http://www.etnassoft.com/2011/02/15/manipulacion-de-arrays-en-javascript/

3. Los objetos

1. Concepto

- En PPO un objeto es una instancia de una clase.
- Una clase es un concepto abstracto sobre algo que posee unas características (propiedades) y una conducta (métodos).
- En JavaScript ya hemos visto que existen clases de objetos predefinidos (Window, Array, String,…) con unas propiedades y métodos ya establecidos.
- Para darles funcionalidad basta con crear instancias de ellos y llamar a sus métodos.
- Pero también se pueden crear objetos de usuario. Es decir objetos personalizados que no existen previamente en el lenguaje.
- El usuario puede declarar un objeto **sin constructor**:

```
Var miObjeto = { propiedad1: 'valor1',  propiedad2: 'valor2' }
```

- Teoría POO Mozilla: https://developer.mozilla.org/es/docs/Web/JavaScript/Introducci%C3%B3n_a_JavaScript_orientado_a_objetos

3. Los objetos

- Para crear una clase propia se sebe crear el constructor.
- Se hace con la instrucción **function**.
- Las **propiedades** se describen por medio del constructor del objeto.
- Utilizando la notación de array con subíndice o array asociativo

```
function persona(valor1, valor2, valor3)
{
        this[0] = valor1;
        this[1] = valor2;
        this[2] = valor3;
}
```

```
function persona(valor1, valor2, valor3)
{
        this[“nombre”] = valor1;
        this[“edad”] = valor2;
        this[“sexo”] = valor3;
}
```

3. Los objetos

2. Constructor

- Utilizando notaciones de nombres

```
function persona(valor1, valor2, valor3)
{
        this.nombre= valor1;
        this.edad= valor2;
        this.sexo= valor3;
}
```

- Un objeto puede ser propiedad dentro de otro objeto

```
function persona(valor1, valor2, valor3, valorObjeto)
{
        this.nombre= valor1;
        this.edad= valor2;
        this.sexo= valor3;
        this.otraPropiedad = otroObjeto
}
```

3. Los objetos

- Instanciar un objeto
  - Una vez creado el constructor puede ser llamado para crear una instancia
  - Para saber si un objeto es de una clase (operador instanceof

```
var  amigo = new persona()
var  amigo = new persona('Juan')
var  amigo = new persona('Juan','30','v')
If(amigo instanceof persona) {alert('es persona')}
```

- Acceso a las propiedades

```
amigo[0] = "Juan"
amigo["nombre"] = "Juan"
amigo.nombre = "Juan"
```

3. Los objetos

- Un método es una función asociada a un objeto
- Se declaran igual que las propiedades pero asociando a funciones.

```
Function persona(nombre, edad, sexo)
{
        this.nombre= valor1;
        this.edad= valor2;
        this.sexo= valor3;
        this.metodo1= function ( ){instrucción;…} ;
        this.metodo2= miFuncion;
}
```

- Ejecución de los métodos
  - Es igual que cualquier función pero va asociada a un nombre de objeto
  - **objeto.metodo()**

3. Los objetos

- Las propiedades y los métodos pueden declararse fuera del constructor con la propiedad **prototype**

```
persona.prototype.decirHola = function() { alert
("Hello"); };
```

3. Los objetos

4. Ejemplo

- Constructor principal:

```
function coche (marca_in, modelo_in, anio_in, concesionario_in)
{ this.marca = marca_in;
  this.modelo = modelo_in;
  this.anio = anio_in;
  this.concesionario = concesionario_in;
  this.imprimedatos = imprimedatos; }
```

- Constructor que hace de propiedad:

```
function concesionario(cod_oficina_in, ciudad_in, responsable_in)
{ this.cod_oficina = cod_oficina_in;
  this.ciudad = ciudad_in;
  this.responsable = responsable_in; }
```

- Función que hace de método:

```
function imprimedatos()
{ document.write("Marca: " + this.marca);
document.write("Modelo: " + this.modelo);
document.write("Año: " + this.anio);
document.write("Concesionario.responsable: "
```

3. Los objetos

- Consultar ejemplos de objetos (this, with):
- http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_JavaScript/OOP
- Consultar ejemplos de la palabra this:
- http://www.etnassoft.com/2012/01/12/el-valor-de-this-en-javascript-como-manejarlo-correctamente/