Desarrollo Web En Entorno Cliente

TEMA 3

Objetos predefinidos en JavaScript

Daniel López Lozano

Escuela Arte Granada

Índice

Objetos nativos (core) del lenguaje.

- String.
- Number.
- Math.
- Array.
- Date.

Índice

Objetos BOM (Browser Object Model).

- Window.
- Screen.
- Location.
- History.
- Navigator.
- setTimer y setInterval.

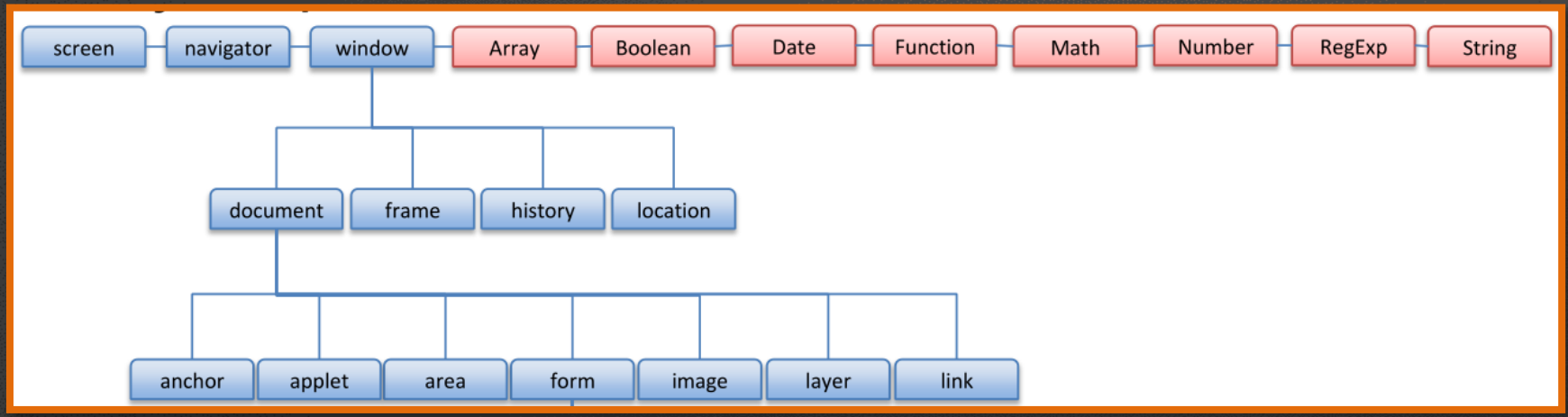
Objetos nativos

Daniel López Lozano

Escuela Arte Granada

- **JavaScript** es un lenguaje **orientado a objetos**.
- Aún no hemos visto **como crear nuestros propios objetos** pero JavaScript facilita una serie de objetos nativos **para simplificar y agilizar tareas**.
- En JavaScript se accede a las propiedades y a los métodos de los objetos **mediante el operador punto** (“.”)
- Los objetos de JavaScript se ordenan de modo jerárquico.

Jerarquía de Objetos en JavaScript



Objeto String

- Las cadenas de **texto son objetos**, pero en JavaScript para facilitar su uso **las considera como tipos primitivos** siempre que las creemos como tal.
- **Se desaconseja totalmente el uso del constructor** ya que puede generar problemas a la hora de comparar distintos strings.
- Crear strings como objetos es menos eficiente y además puede producir errores de comparación como se ha dicho.

Objeto String

- Este tipo de datos existe para dar uniformidad a la orientación a objetos.

```
var x = "Cadena"; // typeof devuelve string
var y = new String("Cadena"); //typeof devuelve object
(x==y) //devuelve true
(x===y) //devuelve false
var a = new String("Cadena");
var b = new String("Cadena");
(a==b) //devuelve false porque son objetos distintos
(a===b) //devuleve true porque son objetos del mismo tipo
```

- El atributo length devuelve la longitud de la cadena.

Objeto String

- Para concatenar dos o más string podemos utilizar el operador + o el método concat().
- La única diferencia es que + puede intentar sumar en lugar de concatenar dependiendo del contexto.
- toUpperCase(): Transforma la cadena a mayúsculas.
- toLowerCase(): Transforma la cadena a minúsculas.
- charAt(posición): Devuelve el carácter de la posición indicada.

Objeto String

- indexOf(caracter): Devuelve la posición en la que se encuentra el string indicado.
- Si el carácter aparece más de una vez, se devuelve su primera aparición.
- Si el carácter no está, devuelve el valor -1.
- lastIndexOf(carácter): Es idéntico a indexOf() salvo que solo se devuelve la última ocurrencia.
- Tiene un parámetro opcional que indica donde empezar a buscar.

Objeto String

```
var mensaje="Hola";
mensaje=mensaje+" mundo"; //guarda "Hola mundo”
mensaje=mensaje.concat(" que tal");
//guarda “Hola mundo que tal”
mensaje="Hola";
mensaje=mensaje.toUpperCase(); //guarda “HOLA”
mensaje=mensaje.toLowerCase(); //guarda “hola”
alert(mensaje.charAt(1)); //muestra o
var mensaje='Casa';
alert(mensaje.indexOf('as')) //muestra 1
alert(mensaje.indexOf('z')) //muestra -1
alert(mensaje.lastIndexOf('a')); //muestra 3
mensaje="prueba prueba";
alert(mensaje.indexOf('p',3) ); //muestra 7
```

Objeto String

- substring(inicio, fin): Devuelve la subcadena entre las posiciones indicadas.
- Si sólo se indica inicio, se devuelve la subcadena desde ese punto hasta el final.
- slice (inicio fin): Idéntica a substring, la única diferencia es que admite índices negativos, si esto es así se empieza a contar desde el final.

Objeto String

- replace(original, cambio): Devuelve una cadena sustituyendo la primera ocurrencia del primer parámetro por el segundo.
- split (separador): Convierte una cadena de texto en un array de cadenas partiendo por la subcadena indicada
- trim(): Devuelve un string sin los espacios extra al principio y al final.

Objeto String

```
var mensaje="Hola mundo";
alert(mensaje.substring(2,6)) //muestra "la mu"
alert(mensaje.substring(5)) //muestra "mundo"
alert(mensaje.slice(-5,-2)) //muestra "mun"
alert(mensaje.slice(-2)) //muestra "do"
mensaje="Usa Explorer";
alert(mensaje.replace("Explorer","Firefox"))
//muestra "Usa Firefox"
mensaje="Hola mundo, mensaje de prueba";
var palabras=mensaje.split(" "); //separa por espacios
var letras=mensaje.split(""); //separa por letras
```

Objeto Number

- Los números en JavaScript se manejan como tipos primitivos, aunque también se pueden crear como objetos.
- Se desaconseja totalmente el uso del constructor ya que puede generar problemas a la hora de comparar distintos números al igual que ocurría con los strings.
- Este tipo de datos admite notación científica y base hexadecimal, octal y binaria.

Objeto Number

- Mediante el método toString() se pueden realizar conversiones a hexadecimal, octal o binario de la siguiente forma:

```
var num = 128;
num=num.toString(16); // devuelve 80
num=num.toString(8); // devuelve 200
num=num.toString(2); // devuelve 10000000
num=num.toString(); // convierte el número a String
```

Objeto Number

- parseInt(x) y parseFloat(x)
- toFixed(x): redondea un Number al número de decimales indicado por parámetro.
- toExponential(x) funciona como toFixed() pero redondea usando notación científica.

```
var x = 9.656;
x.toFixed(0); // devuelve 10
x.toFixed(2); // devuelve 9.66
x.toFixed(4); // devuelve 9.6560
x.toFixed(6); // devuelve 9.656000
```

Objeto Math

- El objeto Math en JavaScript nos permite realizar operaciones matemáticas más avanzadas.
- Math no tiene constructor, no permite crear nuevos objetos.
- Math.PI devuelve el valor de PI, existen otras constantes como E, LN2, LN10, LG2E, LG10E…
- Math.Min() y Math.Max() devuelve el menor de todos los números indicados como parámetro.

Objeto Math

- Math nos proporciona 3 métodos para realizar redondeos:

```
//redondeo tradicional
Math.round(4.4); // devuelve 4
Math.round(4.5); // devuelve 5
//redondeo hacia arriba
Math.ceil(4.2); // devuelve 5
//redondeo hacia abajo
Math.floor(4.9) // devuelve 4
```

Objeto Math

- Math.random(): devuelve un número aleatorio entre 0 (incluido) y 1 excluido.

```
//El intervalo comienza en Y con tamaño X
parseInt(Math.random()*X+Y) ;
//Genera numero entre 0 y X-1
Math.floor(Math.random() * X);

//comparaciones
Math.max(0, 150, 30, 20, -8, -200); // devuelve 150
Math.min(0, 150, 30, 20, -8, -200); // devuelve -200
```

Objeto Math

- Math.sqrt(x): devuelve la raíz cuadrada del número indicado.
- Math.log(x): devuelve el logaritmo del número indicado.
- Math.abs(x): devuelve el valor absoluto del número indicado.
- Math.pow(x,y): devuelve el resultado de elevar x a y

Objeto Array

- Un array se puede crear directamente o usando el constructor.
- Se desaconseja totalmente el uso del constructor ya que puede generar problemas y es menos eficiente.

```
var array = []; o var array=[20];// una sola pos con valor 20
var miArray=[25,"algo", true, 18.7];
var otro = new Array(); // array vacio
var malo=new Array(40); //40 posiciones undefined
var otromas= new Array("lunes", "martes", "miercoles");
```

Objeto Array

- A la hora de mostrar un array, disponemos de distintas opciones

```
alert(array[X]); //muestra la posición X
alert(array); //muestra todo el array separado por comas
alert(array.join()); //muestra todo el array separado por comas
alert(array.join("-")); //muestra todo el array separado por guiones
alert(array.join(" ")); //muestra todo el array separado por espacios
```

- La función devuelve un String con el formato deseado.
- Podemos buscar en un array con indexOf como en String.

Objeto Array

- Podemos añadir fácilmente elementos en la última posición del Array:

```
array[array.length]=X;//Añade x al final del array
```

- Aunque también hay métodos específicos:

```
t=array.push(X);//añade X al final y devuelve el nuevo tamaño
t=array.unshift(X)//añade X al inicio desplazando a todos los demás
```

Objeto Array

- Nunca utilizar el operador + para añadir elementos o concatenar arrays.

```
var array=["a","b","c"];
array=array +"d"; //el nuevo array es “a”,”b”,”cd”;
```

- utilizar el método concat() que es muy potente.

- Permite concatenar nuevos elementos o con otros arrays, incluso permite concatenar varios elementos al mismo tiempo.

Objeto Array

- Ejemplos usando concat():

```
var array=["a","b"];
array=array.concat("c"); // ahora contiene “a”,”b”,”c”
array=array.concat("d","e"); // ahora contiene “a”,”b”,”c”,”d”,”e”

var otro=["Z"];
array=array.concat(otro); // ahora contiene “a”,”b”,”c”,”d”,”e”,”Z”
otro=otro.concat(otro,otro); //ahora tiene “Z”,”Z”,”Z”
```

Objeto Array

- Para extraer elementos de un array JavaScript nos facilita 3 métodos:
- delete(): borra el contenido de la posición indicada dejándola undefined.
- pop(): borra el último elemento del array y lo devuelve (reduciendo el tamaño).
- shift(): igual que pop pero por el principio.

Objeto Array

- Ejemplos usando métodos de extracción:

```
var array=["a","b","c"];
delete array[1]; // borra la “b” y deja esa posición undefined
var x=array.pop(); // x contiene ”c” y el array tiene tamaño 2
var x=array.shift(); // x contiene ”a” y el array tiene tamaño 1
```

Objeto Array

Otros métodos interesantes:

- slice(): es un método que permite extraer un subconjunto del array.
- reverse(): modifica el array original invirtiendo el orden de sus elementos.
- sort(): modifica el array ordenando alfabéticamente.

Objeto Array

- Ejemplos:

```
var array=["a","b","c"];
array.slice(x,y); //devuelve desde la posición x (incluida) hasta la y (excluida)
array.slice(x);//devuelve desde la posición x hasta el final

array.reverse();
alert(array); //muestra “c”,”b”,”a”

array.sort ();
alert(array); //muestra “a”,”b”,”c”
```

Objeto Array

- EJERCICIO 1: Realiza una función que pida 5 números los almacene en un array y los muestre ordenados.
- EJERCICIO 2: Modifica el ejercicio anterior para que estos 5 números se pidan en un solo prompt.
- EJERCICIO 3: Realiza un script que genere 5 números aleatorios y los meta en un array.
- EJERCICIO 4: Igual que el anterior pero asegurando que los números sean distintos.

Objeto Date

- La clase Date nos permite trabajar con fechas y horas e incorpora un amplio número de métodos para manipularlas.
- Una fecha puede mostrarse como un String “Sat Oct 3 2015 12:30:45 GMT+0200 “
- Como un entero 1443868245360 que son los milisegundos que han transcurrido desde el 1 de Enero de 1970 a las 00:00:00
- Para crear una fecha se usa el constructor new Date()

Objeto Date

```
new Date() //crea un objeto fecha con la hora y fecha actuales
new Date(milisegundos) //crea la fecha indicada en milisegundos
new Date(String) //Indicada como texto ej: new Date("October 3, 2015 12:35:00");
new Date(año, mes, dia, hora, minuto, segundo, millisegundo) //mínimo 3 enteros
```

- Muy importante el mes se cuenta desde 0 que es Enero hasta 11 que es Diciembre.
- Podemos especificar una nueva fecha de alguna de las siguientes formas:
- Fecha larga “MMM DD AAAA” =>“Oct 03 2015”

Objeto Date

- Fecha corta MM/DD/AAAA ó AAAA/MM/DD
  “10/03/2015” ó “2015/10/03”
- Dos objetos Date pueden compararse directamente con los operadores relacionales <,<=,>,>=,==

```
new Date("2015-10-3"); //más utilizado
new Date("2015-10"); // por defecto día 1
new Date("2015"); // por defecto 1 de Enero
new Date("2015-10-3T12:38:21");
```

- JavaScript ignorará errores y completará las fechas

Objeto Date

- Métodos selectores del objeto Date

```
getDate() //Devuelve el día del mes como número (1-31)
getDay() //Devuelve el día de la semana en número (0-6) 0=Domingo
getMonth() //Devuelve el mes como entero, comenzando desde 0
getFullYear() //Devuelve el año con 4 dígitos (YYYY)
getHours() //Devuelve la hora (0-23)
getMinutes() //Devuelve el minuto (0-59)
getSeconds() //Devuelve el segundo (0-59)
getMilliseconds() //Devuelve el milisegundo (0-999)
getTime() //Devuelve los milisegundos desde el 1 de Enero de 1970
getTimezoneOffset() //Devuelve el desfase horario con GMT en minutos
```

Objeto Date

- Métodos modificadores del objeto Date

```
setDate(día) //Modifica el día del mes (1-31)
setMonth(mes) //Modifica el mes (entero desde 0)
setFullYear(año) //Modifica el año
setFullYear(año, mes, día) //Modifica la fecha completa (mes desde 0)
setHours(h) //Modifica la hora (0-23)
setMinutes(m) //Modifica el minuto (0-59)
setSeconds(s) //Modifica el segundo (0-59)
setMilliseconds(m) //Modifica el milisegundo (0-999)
setTime(m) //Modifica los milisegundos sumando al 1 Enero de 1970
```

Objeto Date

- Si al modificar damos un valor mayor este se reajusta automáticamente.
- Por lo que es posible avanzar y retrasar fechas en base a cualquier unidad.

```
var fecha=new Date("2015-09-28");
alert(fecha);
fecha.setDate(fecha.getDate()+7);
alert(fecha);
fecha.setMonth(fecha.getMonth()+5);
alert(fecha);
fecha.setHours(55);
alert(fecha);
fecha.setMinutes(75);
alert(fecha);
```

Objeto Date

- EJERCICIO 5: Crear un script muestre la fecha un dia anterior y posterior a la fecha actual.
- EJERCICIO 6: Crea una función que reciba una fecha en la que se da de baja una persona y el número de días de la baja. El script debe usar la función cogiendo datos de entrada y mostrar un mensaje “Fecha de regreso: …“

Objetos BOM

Daniel López Lozano

Escuela Arte Granada

- El BOM (Browser Object Model) permite a JavaScript “comunicarse” con el navegador para obtener información o modificar aspectos de este.
- No existe un estándar que regule el BOM, por lo que existen diferencias entre los navegadores en los cuales se ejecute nuestro código.
- A pesar de esto, los navegadores actuales soportan la mayoría de los métodos o propiedades para interactuar con JavaScript.

Objeto window

- El objeto window es soportado por TODOS los navegadores y representa la ventana del navegador.
- Los navegadores actuales permiten abrir nuevos documentos HTML en nuevas ventanas o pestañas dentro de la misma ventana del navegador.
- Cada nueva ventana o pestaña se corresponde con un objeto window diferente.

Objeto window

- Los principales métodos y atributos que dependen de window tienen que ver con el tamaño y la posición de la ventana del navegador, así como abrir y gestionar nuevas ventanas.
- Los atributos y métodos del objeto Window son accesibles directamente sin necesidad de usar el punto.

window.alert(“Hola”)

- Para cuestiones relacionadas con el manejo de ventanas sí usamos window.(atributo o método)

Objeto window

- Para el saber tamaño de las ventanas tenemos las propiedades window.innerHeight y window.innerWidth.
- Existe el método open permite abrir nuevas ventanas, recibiendo tres parámetros de tipo string:

open(“url”,”nombre”,”propiedades”);

- Url: Carga en la nueva ventana la url indicada.
- Nombre: nombre de la ventana (no es el titulo).
- Propiedades: Lista de propiedades de la ventana.

Objeto window

- toolbar: Indica si la ventana tendrá barra de herramientas o no.
- location: Indica si la ventana tendrá campo de localización (url) o no.
- directories: Indica si la ventana tendrá botones de dirección o no.
- status: Indica si la ventana tendrá barra de estado o no.
- menubar: Indica si la ventana tendrá barra de menús o no.
- scrollbars: Indica si la ventana tendrá barras de desplazamiento o no.
- resizable: Indica si la ventana se puede redimensionar o no.
- width: Indica el ancho de la ventana en pixels (sólo se indica el número).
- height: Indica el alto de la ventana en pixels (sólo se indica el número).
- left: Indica los pixels desde el lado izquierdo donde se abrirá la ventana.
- top: Indica los pixels desde el lado superior donde se abrirá la ventana.

Objeto window

- Ejemplo
  open(“pagina.html”, “nueva”, “menubar=no, titlebar=yes”);

- Si al abrir una ventana con open la guardamos en una variable, después la podemos referenciarla para aplicar métodos o atributos a windows.

- Si no, estos métodos se aplicarán siempre sobre la ventana actual.

Objeto window

- Ejemplo

```
open('url', 'nombre', 'propiedades');
document.write("hola"); //ambos mensajes se escriben en la ventana original
window.document.write("mundo");

var nueva=open('url', 'nombre', 'propiedades');
nueva.document.write("hola"); //ahora se escriben en la ventana nueva
nueva.document.write("mundo");
```

- EJERCICIO 7: Crea una página que pida un mensaje para incluir en una nueva ventana. Después creará una nueva ventana y escribiendo el mensaje leído en la nueva ventana ya sea por alert, write...

Objeto window

Propiedades interesantes del objeto window:

- closed: Dice si la ventana está cerrada (true) o no (false).
- history: Array con las las URLS visitadas (almacenadas en su historial).
- location: Cadena con la URL de la barra de dirección.
- name: Nombre de la ventana.
- status: String con el mensaje que tiene la barra de estado.
- top: Referencia a la ventana padre. (como super en java)

Objeto window

- Métodos interesantes del objeto window:
- blur(): Le quita el foco al objeto window.
- close(): Cierra el objeto window.
- focus(): Pone el foco en el objeto window.
- moveBy(x,y): Mueve el objeto window los pixeles especificados en (x,y).
- moveTo(x,y): Mueve el objeto window a las coordenadas (x,y).
- resizeTo(x,y): Redimensiona el objeto window al tamaño indicado (x,y).
- scroll(x,y): Desplaza el objeto window a las coordenadas especificadas.
- scrollBy(x,y): Desplaza el objeto window los pixels especificados.

Objeto Screen

- Hace referencia a la pantalla del usuario. Solo permite obtener alguna información para las ventanas del navegador.
- screen.width: ancho de la ventana del usuario en píxeles.
- screen.height: alto de la ventana del usuario en píxeles.
- screen.availWidth: ancho útil (restando elementos de interfaz)
- screen.availHeight: alto útil (restando elementos de interfaz)
- screen.colorDepth: número de bits para mostrar el color
- screen.pixelDepth: equivalente al anterior en ordenadores actuales

Objeto Location

- El objeto location permite recuperar la dirección de la página actual (URL) y redireccionar el navegador a nuevas páginas.
- hostname: Nombre de dominio del servidor (o la dirección IP).
- href: Cadena que contiene la URL completa.
- pathname: Cadena que contiene el camino al recurso de la URL.
- port: Cadena que contiene el número de puerto del servidor
- protocol: Cadena que contiene el protocolo utilizado.
- reload(): Vuelve a cargar la URL especificada en la propiedad href
- assign('nueva'): Carga la nueva URL indicada.

Objeto History

- El objeto history hace referencia al historial del navegador.
- Para proteger la privacidad de los usuarios, las opciones de JavaScript sobre el historial son muy limitadas:
- history.back(): Tiene una funcionalidad idéntica a la ir a la página anterior del navegador.
- history.forward(): Tiene una funcionalidad idéntica a la de ir a la página siguiente del navegador.

Objeto Navigation

- Contiene información relativa al navegador del cliente.
- appCodeName: Contiene el nombre del navegador *.
- appName: Contiene el nombre del navegador *.
- appVersion: Contiene información sobre la versión del navegador.
- cookieEnabled: Indica si el navegador tiene activas las cookies (true/false)
- language: Cadena de dos caracteres con el idioma del navegador.
- platform: Contiene el sistema operativo del cliente*.
- product: Contiene el motor del navegador
- userAgent: Contiene toda la información del navegador del cliente.
- javaEnabled(): Indica si el navegador tiene activado java (true/false)

setTimer y setInterval

- El objeto window permite la ejecución de código programado en intervalos de tiempo.
- Estos métodos son conocidos comúnmente como temporizadores o eventos temporizados
- setTimeout(funcion, retardo) permite ejecutar una función tras un retardo determinado, este retardo se expresa en milisegundos.
- Una vez consumido el tiempo la función se ejecuta de manera normal.

setTimer y setInterval

```
function tiempo10(){alert("han pasado 10 seg")};
function tiempo3(){alert("han pasado 10 seg")};
setTimeout(tiempo10 , 10000); //tras 10 segundos muestra el alert
setTimeout(tiempo3 , 3000); //tras 3 segundos muestra el alert
```

- EJERCICIO 8: Crear una pagina, que al llevar 5 segundos en ella nos abra una nueva ventana a una página con el mensaje “Anuncio de publicidad” y una imagen cualquiera. Pon el foco en esta página nueva y tras llevar 10 segundos abierta ciérrala.

setTimer y setInterval

- setInterval(funcion, retardo) es similar a la anterior, pero en este caso la función se repite indefinidamente cada vez que se consume la pausa indicada.
- Tanto con setTimeout como con setInterval, la ejecución del script continua mientras se calcula la pausa y cuando esta se consume se ejecuta la función.
- Es decir, el script no se “congela” hasta que se completa la pausa, sino que sólo se pospone la ejecución de la función indicada.

setTimer y setInterval

- También es posible detener estas funciones antes de que se ejecuten o en cualquier momento borrando los temporizadores.
- Para esto existen las funciones clearTimeout() y clearInterval().
- Es necesario guardar los temporizadores en una variable, la cual habrá que indicarle a clearTimeout y clearInterval para que sepa que contador de tiempo detener.

setTimer y setInterval

```
function anuncio(){alert("han pasado 3 segundos");}
//muestra el alert cada 3 segundos.
setInterval(anuncio, 3000);
```

```
function pierde(){alert("has perdido");}
function gana(){alert("has ganado");}
var primero=setTimeout(pierde, 3000);
var segundo=setInterval(gana, 5000);
clearTimeout(primero);
clearInterval(segundo);
```

- EJERCICIO 9: Un script que cree una ventana y haga que se mueva de izquierda a derecha cada 5 segundos hasta que salga por la pantalla.
- EJERCICIO 10: Igual que el anterior pero que cambie el tamaño cada 5 segundos.
- EJERCICIO 11: Haz un script que muestre una ventana en una posición aleatoria cambiándola cada 5 segundos.
- EJERCICIO 12: Crea una función que se ejecute cada 2 segundos y que muestre el mensaje “Soy un anuncio” más el valor de un contador, cada vez que el mensaje se muestre se incrementará el contador. El intervalo se detendrá cuando el anuncio se haya mostrado 5 veces. (Pista: variables globales)

Fin del Tema 3

Bibliografía

- Gauchat, Juan Diego: **“El gran libro de HTML5, CSS3 y JavaScript”**.Editorial Marcombo. 2012
- Vara, J.M. y otros: **“Desarrollo Web en Entorno Cliente. CFGS”**.Editorial Ra-Ma. 2012
- **“Programación en JavaScript”.** Colección de artículos disponibles en la url. http://www.desarrolloweb.com/manuales/ Última visita: Septiembe 2017.
- **W3SCHOOL “Manual de referencia y Tutoriales”** http://www.w3schools.com/ Última visita Septiembre 2017
- **Comesaña, J.L. : Técnico en Desarrollo de Aplicaciones Web.** http://www.sitiolibre.com/daw.php Última visita Septiembre 2017