

UD 03. OBJETOS PREDEFINIDOS

Índice de contenido

1.	Introducción.....	2
2.	Funciones predefinidas.....	2
2.1	Tratamiento numérico	2
2.2	Función eval	2
3.	Objeto predefinido String	3
4.	Objeto predefinido Date.....	5
5.	Objeto predefinido Array	5
6.	Objeto predefinido Math	6
7.	Objeto predefinido Window	8
8.	Funciones anónimas.	9
9.	Material adicional	9
10.	Bibliografía	9



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

TEMA 3 OBJETOS PREDEFINIDOS

1. INTRODUCCIÓN

Entre otras posibilidades, Javascript es un lenguaje que de forma nativa posee gran cantidad de funciones y objetos predefinidos.

Estas funciones y objetos nos pueden ser útiles para realizar un código más eficiente, claro y ahorrarnos tiempo reinventando la rueda.

A continuación, describiremos algunos de los objetos y funciones predefinidas más importantes.

2. FUNCIONES PREDEFINIDAS

2.1 Tratamiento numérico

Hay dos funciones predefinidas (`parseInt` y `parseFloat`) que ya vimos en la unidad anterior. Además de recordarlas, ampliamos con la función `isNaN` que nos ayudará a distinguir si una cadena es un número o no.

- `parseInt(cadena)`: convierte una cadena de texto entero a entero.
- `parseFloat(cadena)`: convierte una cadena de texto a decimal.
- `isNaN(cadena)`: NaN es la abreviación de “Not a Number”. Esta función comprueba si una cadena de caracteres puede ser considerada un número (da false) o no (da true).

Ejemplo:

```
do{  
    var numero=prompt("Esto se repetirá hasta que metas un número");  
}while(isNaN(numero));
```

2.2 Función eval

Eval es una función que recibe una cadena y la interpreta como código Javascript. También vale para expresiones numéricas, donde te devuelve el resultado.

Es una función muy útil ya que podemos construir código dinámicamente mediante una cadena. Es una función a utilizar únicamente en situaciones que lo requiera, ya que una cadena interpretada por eval que esté formada maliciosamente puede causar muchos inconvenientes.

La sintaxis es `eval(cadena)`;


Ejemplo:

```
var x=3;  
var y=2;  
var a=eval("2+3");  
var b=eval("x*y");  
  
eval("alert('a vale '+a+' b vale '+b)");
```

3. OBJETO PREDEFINIDO STRING

Cuando creamos una cadena, implícitamente esta cadena se convierte en un objeto String, con sus propiedades y métodos predefinidos.

El objeto predefinido String es útil porque nos ayuda a realizar múltiples operaciones con cadenas.

 **Muy importante:** los métodos de cadenas de Javascript, no modifican al objeto actual, sino que devuelven el resultado de aplicar una modificación.

Si queremos que la modificación se aplique sobre la misma cadena que estamos trabajando, haremos algo así:

```
cadena=cadena.metodoQueDevuelveUnaModificacion();
```

En este objeto, hay una propiedad muy importante que es `length`. Esta propiedad te dice cuántos elementos (caracteres) tiene la cadena.

También existen una serie de métodos muy útiles.

- `toLowerCase()/toUpperCase()`: devuelve la cadena convertida a minúsculas/mayúsculas.
- `concat(cadena)`: devuelve el objeto con el valor de cadena concatenado al final.
- `charAt(posicion)`: devuelve el carácter que se encuentre en la posición solicitada. Debemos tener en cuenta que las posiciones comienzan a contar desde cero.
- `indexOf(texto, [indice])`: devuelve la primera posición donde se encuentra el texto buscado, empezando a buscar desde la posición "índice". Si índice no se indica, se toma por defecto el valor 0.
- `lastIndexOf (texto, [indice])`: como la anterior. Busca "hacia atrás" la primera ocurrencia del texto buscado. Índice indica desde que punto se empieza a buscar "hacia atrás". Si no se indica el valor de índice, se busca desde el final.
- `replace(texto1,texto2)`: busca ocurrencias de la cadena texto1 y las reemplaza por texto2.
- `split(caracter, [trozos])`: separa la cadena mediante un carácter separador. Trozos indica el máximo de separaciones. Si no se indica, se harán todas las separaciones posibles.
- `substring(inicio, [fin])`: devuelve la sub cadena comprendida entre la posición inicio y la posición fin. Si fin no se indica, se toma como valor el final de la cadena.

Ejemplo de algunas de las funciones

```
var cad="Pascual:Perez:123456";
var tfo;
cad=cad.toUpperCase();
alert(cad);
splitTodosCampos=cad.split(":");
split1Campo=cad.split(":",1);
alert(splitTodosCampos);
alert(split1Campo);
tfo=splitTodosCampos[2];
//Cambio en el telefono los números 3 por 9s
tfo=tfo.replace("2","9");
alert(tfo);
```

```
//Muestro el quinto número del teléfono  
alert(tfo.charAt(4));  
alert("Bienvenido al CPIFP Bajo Aragón");
```

4. OBJETO PREDEFINIDO DATE

Date es un objeto predefinido que nos permite trabajar con fechas. Para crear un objeto date se admiten múltiples formatos. Más información en http://www.w3schools.com/js/js_dates.asp

Nota: los meses en Date se numeran del 0 al 11 (siendo el 0 Enero y el 11 Diciembre) mientras que los días si se numeran del 1 al 31.

Ejemplo:

```
// Crea una fecha con la fecha y hora del sistema
var d=new Date();
// Crea una fecha basandose en la cadena de fecha especificada
d=new Date("October 13, 2014 11:13:00");
// Crea una fecha indicando año, mes, día, horas, minutos, segundos milisegundos
d=new Date(99,5,24,11,33,30,0);
// Crea una fecha indicando año, mes, día.
d=new Date(99,5,24);
```

Algunos de los métodos más importantes del objeto Date:

- Métodos set/get: son métodos que permite cambiar el valor de alguna parte de la fecha (set) u de obtener el valor de alguna parte de la fecha (get).
 - Ejemplos: setMonth(mes), getMonth(); setDate(día), getDate(), setHours(hora, minuto,segundo), getHours(), etc.
- getDay(): devuelve el día de la semana, (el día del mes es con getDate). Están numerados del 0 al 6, siendo el 0 domingo y el 6 sábado.
- toString(): convierte la fecha del objeto a una cadena en objeto fecha.
- toGMTString(): convierte la fecha del objeto en una cadena con formato de fecha GMT.
- toUTCString(): convierte la fecha del objeto en una cadena con formato de fecha UTC.

Ejemplo:

```
// Con fecha actual y con día uno del mes, mostramos día de la semana
var d=new Date();
alert(d.getDay());
d.setDate(1);
alert(d.getDay());
```

5. OBJETO PREDEFINIDO ARRAY

Un array es un objeto para el manejo de arrays que nos puede ser muy útil para realizar determinadas operaciones. En un array se pueden guardar cualquier tipo de objetos (date, string, números enteros, decimales, otros Arrays, etc....).

Todo lo aplicado al uso de arrays que comentamos en el tema anterior, es aplicable al objeto Array (length para número de elementos, modificar/consultar array, etc....).



Generalmente los métodos aplicados en el Array modifican el objeto original.

Algunos de los métodos más importantes del objeto Array:

- `join([separador])`: devuelve una cadena con todos los elementos del array, separados por el texto que se incluya en `separador`.
- `push(elemento1, elemento2, ...)`: añade al final los elementos 1 y 2 en el orden proporcionado.
- `pop()`: devuelve y elimina el último elemento del array.
- `reverse()`: invierte el orden de los elementos de un array.
- `sort()`: ordena los elementos de un array alfabéticamente. Nota: al ser un orden alfabético, puede dar resultados incorrectos con números, por ejemplo 10 ir antes que 2.
- `slice(inicio, [final])`: devuelve los elementos de un array comprendidos entre `inicio` y `final`. Si no se indica `final`, se toma hasta el último elemento del array.

Ejemplo:

```
var x=[1,2,3];
var y=[2,3,10,2];
// Damos la vuelta a un vector
x.reverse();
alert(x);
// Ordenamos (recordamos que es alfabético y no numérico)
y.sort();
alert(y);
// Sacamos y eliminamos elemento
var sacado=y.pop();
alert(sacado);
alert(y);
```

6. OBJETO PREDEFINIDO MATH

El objeto Math es un objeto que contiene una colección de métodos que nos ayudarán a trabajar realizando operaciones aritméticas, redondeos, etc.

Algunas de las constantes matemáticas predefinidas más importantes son:

- `E`: almacena el número de Euler.
- `PI`: almacena el número Pi.
- `LN2` / `LN10` : logaritmo neperiano de 2 / 10.
- `LOG2E` / `LOG10E` : logaritmo en base 2 / 10 del número de Euler.

Algunos de los métodos más importantes que tienes:

- Redondeo:
 - `floor(numero)`: redondea hacia abajo un número.
 - `ceil(numero)`: redondea hacia arriba un número.
 - `round(numero)`: redondea al entero más cercano.
- Matemáticos:

- `abs(numero)`: devuelve el número en valor absoluto.
- `max / min (x,y)`: devuelve el mayor / menor de dos números x e y.
- `pow(x,y)`: devuelve x elevado a y.
- `random()`: devuelve un número aleatorio con decimales entre 0 y 1.
- `sqrt(numero)`: devuelve la raíz cuadrada del número indicado.

Ejemplo:

```
// Obtenemos un entero entre 1 y 11
var x=parseInt( (Math.random()*10)+1 )
// Redondeamos y hacia abajo
var y=Math.floor(11.5);
alert (Math.PI);
```

7. OBJETO PREDEFINIDO WINDOW

El objeto Window es un objeto que tiene propiedades y controla elementos de lo que ocurre en la “ventana” del navegador.

Los métodos que estudiamos en el tema anterior como alert, prompt, etc. forman parte del objeto Window. Para hacer llamada a estos métodos no hace falta nombrar explícitamente Window (el navegador ya se encarga de ello).

Algunos de los métodos más importantes no estudiados previamente son:

- `setTimeout(cadenaFuncion, tiempo)`: este método ejecuta la llamada a la función proporcionada por la cadena (se puede construir una cadena que lleve parámetros) y la ejecuta pasados los milisegundos que hay en la variable tiempo. Devuelve un identificador del “setTimeout” que nos servirá para referenciarlo si deseamos cancelarlo. SetTimeout solo ejecuta la orden una vez.
- `setInterval(cadenaFunción, tiempo)`: exactamente igual que setTimeout, con la salvedad de que no se ejecuta una vez, sino que se repite cíclicamente cada vez que pasa el tiempo proporcionado.
- `clearTimeout / clearInterval (id)`: se le pasa el identificador del timeout/interval y lo anula.

Ejemplo:

```
// Creamos un intervalo que cada 15 segundos muestra mensaje hola
Var idA=setInterval("alert('hola');",15000);
// Creamos un timeout que cuando pasen 3 segundos muestra mensaje adios
Var idB=setTimeout("alert('adios');",3000);
// Creamos un timeout que cuando pasen 5 segundos muestra mensaje estonoseve
Var idC=setTimeout("alert('estonoseve');",5000);
// Cancelamos el ultimo timeout
clearTimeout(idC);
```


8. FUNCIONES ANÓNIMAS.

Cuando se define una función, a menudo se le asigna un nombre y luego se invoca con ese nombre, de esta manera:

```
// Función normales
foo();

function foo(){
    //codigo
}
```

Cuando define una función de esta manera, el tiempo de ejecución de Javascript almacena su función en la memoria y luego crea una referencia a esa función, utilizando el nombre que le ha asignado. Ese nombre es entonces accesible dentro del alcance actual. Esta puede ser una forma muy conveniente de crear una función, pero Javascript no requiere que asigne un nombre a una función. Lo siguiente también es perfectamente legal:

```
// Función anonimas

(function (){
    alert('Esta función no tiene nombre y se ejecutara inmediatamente');
})();
```

Cuando una función se define sin un nombre, se conoce como una función anónima. La función se almacena en la memoria, pero el tiempo de ejecución no crea automáticamente una referencia a la misma para usted. A primera vista, puede parecer que tal cosa no tendría ningún uso, pero hay varios escenarios donde las funciones anónimas son muy convenientes.

9. MATERIAL ADICIONAL

[1] Curso de Javascript en Udacity <https://www.udacity.com/course/javascript-basics--ud804>

10. BIBLIOGRAFÍA

[1] Referencia Javascript

<http://www.w3schools.com/jsref/>

[2] Universidad de Sevilla: Objetos predefinidos Javascript

<http://www.cs.us.es/cursos/mp/temas/Web-tema-09.pdf>

[3] Instituto tecnológico Celaya : objetos predefinidos en Javascript

http://sisinfo.itc.mx/ITC-APIRGG/Fundamentos_PHP/JavaScript/cap7.htm