Desarrollo Web en Entorno Cliente

UD 03. Objetos predefinidos

Parte 1

Actualizado Octubre 2023

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

UD03. 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 evitando "reinventar 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 comentamos en la unidad anterior. En este punto, 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 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 (false) o no (true).

Ejemplo:

```
let numero;
do{
      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. Dado que Javascript admite expresiones numéricas, se puede usar eval para calcular el resultado de expresiones numéricas.

Eval es una función muy útil ya que podemos construir código dinámicamente mediante una cadena. Pero… al ser una función muy poderosa, también es muy peligrosa. Debe utilizarse únicamente en situaciones que lo requiera, ya que una cadena interpretada por eval que esté formada maliciosamente puede causar un agujero de seguridad.

> ❗ **Atención:** esta función se utilizaba para procesar JSON. Actualmente por seguridad se utilizan otras funciones para procesarlo. Virus

Ejemplo:

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

// en este caso eval calcula la suma y la multiplicación aunque sean texto.
```

3. OBJETO PREDEFINIDO STRING

> ❗ **Atención:** aunque Javascript no exige la declaración de tipos de datos, internamente sus variables sí que tienen un tipo de datos concreto.

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.

> ⍰ **Importante:** los métodos de manipulación de cadenas de Javascript, no modifican al objeto actual, sino que devuelven el objeto resultante de aplicar la modificación.

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

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

En el objeto string, hay una propiedad muy usada llamada **"length".** Esta propiedad nos indica cuantos 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, índice**): 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, [índice])**: como la anterior. Busca "hacia atrás" la primera ocurrencia del texto buscado. Índice indica desde qué 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 subcadena 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:

```
let cad="Sergi:Garcia:123456";
let tfo;
cad=cad.toUpperCase();
alert(cad);
let splitTodosCampos=cad.split(":");
// corta todos los campos
let split1Campo=cad.split(":",1);
// solo corta una vez, solo corta Sergi
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));
```

4. OBJETO PREDEFINIDO DATE

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

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
let d=new Date();
// Crea una fecha basándose en la cadena de fecha especificada
d=new Date("October 13, 2014 11:13:00");
// Crea una fecha indicando año, mes, dia, 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 permiten cambiar el valor de alguna parte de la fecha (set) o de obtener el valor de alguna parte de la fecha (get).
    - Ejemplos: setMonth(mes), getMonth(); setDate(dia), 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.
- **toDateString()**: 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 dia uno del mes, mostramos dia de la semana

```
let d=new Date();
alert(d.getDay());
d.setDate(1);
alert(d.getDay());
```

5. OBJETO PREDEFINIDO ARRAY

En la unidad anterior describimos que es un array en Javascript. Para el uso de arrays, Javascript utiliza un objeto que nos puede ser muy útil para realizar determinadas operaciones.

Aclarar que en el interior de 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...).

> **Importante:** generalmente, los métodos del objeto predefinido Array modifican el contenido del propio array.

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])**: corta los elementos de un array comprendidos entre inicio y final. Si no se indica final, se toma hasta el último elemento del array.

Ejemplo:

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