UNIDAD 2.

Objetos predefinidos del lenguaje

Desarrollo Web en Entorno Cliente

2º DAW

Contenidos

Introducción a los objetos

Tipos de objetos que existen:

- **OBJETOS NATIVOS** (no dependen del navegador)
  - String
  - Date
  - Number
  - Math
  - Boolean
  - RegExp
  - Array
  - Function
  - Object
- **OBJETOS DE ALTO NIVEL** (dependen del navegador)
  - Asociados al navegador (BOM, *Browser Object Model* o modelo de objetos del navegador)
    - window
    - navigator
    - screen
    - history
    - location
  - Asociados al documento (DOM, *Document Object Model* o modelo de objetos del dominio)
    - document
- **OBJETOS DEFINIDOS POR EL USUARIO**
  - Cómo crear un objeto (con propiedades)

    ```
    var a = {prop: valor, prop: valor, …}
    ```

    Ejemplo:

    ```
    var animal = {nombre: "pipo", tipo: "gato"};
    ```

Objetos. Constructores integrados

Se pueden crear variables que contengan objetos que sean instancias de los objetos nativos. En general, no es práctico y, por lo tanto, no recomendable crear objetos de tipo **`String`**, **`Number`**, **`Boolean`**, **`Array`**, **`RegExp`**, **`Function`**, **`Object`** con **`new()`**:

En lugar de:	Es mejor:
`var a = new String("cadena");`	`var a = "cadena";`
`var a = new Number(5);`	`var a = 5;`
`var a = new Boolean(true);`	`var a = true;`
`var a = new Array();`	`var a = [1, 2, "hola"];`
`var a = new RegExp();`	`var a = /…/;`

`var a = new Function();`	`var a = function() {...};`
`var a = new Object();`	`var a = {prop = valor, prop = valor…};`

Excepciones:

- **`Math`** no se puede instanciar porque es un objeto global y no tiene constructor. Como es un objeto estático, podemos utilizar sus propiedades y métodos sin crear un objeto de este tipo.
- **`Date`**: el único modo de crear fechas es instanciar este objeto:

```
var a = new Date(...);
```

Objetos nativos: String

PROPIEDADES

Propiedad	Contiene
***cadena*`.length`**	Longitud de la cadena.

MÉTODOS DE BÚSQUEDA

Método	Devuelve
***cadena*`.charAt(`*num*`);`**	Carácter que está en la posición *num.*
***cadena*`.indexOf(`*car*`);`**	Posición de la primera aparición del carácter *car.*
***cadena*`.lastIndexOf(`*car*`);`**	Posición de la última aparición del carácter *car.*
***cadena*`.search(`*cad*`|`*exp*`);`**	Posición de la primera aparición de la subcadena *cad*, o de una cadena que coincida con la expresión regular *exp.*
***cadena*`.match(`*exp*`);`**	Posición de la primera aparición de una subcadena que coincida con la expresión regular *exp.*

MÉTODOS DE COMPARACIÓN

Método	Devuelve
***cadena*`.localeCompare(`*cad*`);`**	-1 si *cadena* es menor que el parámetro *cad.* 0 si *cadena* es igual que el parámetro *cad.* +1 si *cadena* es mayor que el parámetro *cad.* (para determinar si es mayor, igual o menor, se utiliza el orden alfabético A < Z).

MÉTODOS PARA SABER SI UNA CADENA INCLUYE, EMPIEZA O TERMINA CON OTRA

Método	Devuelve
`cadena.includes(cad);`	true, si *cadena* incluye el parámetro *cad.*
`cadena.startsWith(cad);`	true, si *cadena* comienza con el parámetro *cad.*
`cadena.endsWith(cad);`	true, si *cadena* termina con el parámetro *cad.*

MÉTODOS PARA OBTENER UNA CADENA A PARTIR DE OTRA

Método	Devuelve
`cadena.concat(cad);`	Cadena formada por *cadena* seguida de *cad.*
`cadena.repeat(num);`	Cadena formada por *cadena* repetida *num* veces.

MÉTODOS PARA EXTRAER PARTE DE UNA CADENA

Método	Devuelve

`cadena.slice(`
`  pos_ini,`
`  [pos_fin]);`
	Subcadena que comienza con *pos_ini* y termina en *pos_fin:*
	– Si *pos_ini* es un número negativo, se empieza desde el final de la cadena.
	– Si se omite *pos_fin*: se toma hasta el final.

`cadena.substring(`
`  pos_ini,`
`  pos_fin);`
	Similar al anterior, pero ambos parámetros son obligatorios:
	– Si pos_ini es un número negativo, considera que es 0.
	– Si pos_ini > pos_fin, intercambia sus valores.

`cadena.substr(`
`  pos_ini,`
`  [long]);`
	Similar a *slice* pero cambia el segundo parámetro. Subcadena que comienza con *pos_ini* y tiene una longitud de *long*;
	– Si pos_ini es un número negativo, se empieza desde el final de la cadena.
	– Si se omite long: se toma hasta el final.

`cadena.split(`
`  car,`
`  [num_veces]);`
	Array formado por las subcadenas que se obtienen al dividir la *cadena* utilizando como separador el carácter *car*, el número de veces *num_veces*.

`cadena.trim();`	Cadena igual a la *cadena*, eliminando los espacios al principio y al final.

MÉTODOS PARA CONVERTIR A MAYÚSCULAS Y MINÚSCULAS

Método	Devuelve
`cadena.toLowerCase();`	Cadena con los caracteres convertidos a minúsculas.
`cadena.toUpperCase();`	Cadena con los caracteres convertidos a mayúsculas.

MÉTODOS ESPECIALES

Método	Devuelve
***cadena*.toString();**	Devuelve una cadena que representa al objeto. Todos los objetos tienen este método. Lo usamos implícitamente cuando el objeto se representa como un valor de texto o cuando un objeto se referencia de tal manera que se espera una cadena.
***cadena*.valueOf();**	Valor primitivo de la cadena. Todos los objetos tienen este método. Se usa implícitamente cuando hay un objeto y se esperaba un valor primitivo.

Objetos nativos: Number

Solo hay un tipo de datos numérico, que siempre es almacenado como números en punto flotante de doble precisión (64 bits).
Los números se pueden representar en decimal (enteros, con decimales, con parte exponencial), en hexadecimal… ej. 34, 34.05, 123e4, 123e-4, 0xFF.

VALORES ESPECIALES DE TIPO *number*

Valor	Se puede obtener	Ejemplo
`Infinity`	n / 0	`4/0`
`-Infinity`	- n / 0	`-4/0`
`NaN`	*número <operador_distinto_de_+> cadena_no_convertible_a_número*	*`4 - "3 hola"`*

FUNCIÓN isNaN()
Devuelve un valor Boolean. Esta función es necesaria porque, aunque una expresión se puede comparar con `Infitity` y `-Infinity` para saber si una variable ha adquirido ese valor, no se puede comparar con `NaN`, para determinar tiene valor `NaN` o no, ya que `NaN == NaN` y `NaN === NaN` se evalúan a `false`. Por eso, es necesaria esta función.

Función	Devuelve
isNaN(exp);	Devuelve *true* si la *exp* tiene valor `NaN`. La función intenta convertir *exp* a un número. Si el parámetro no se puede convertir, devuelve *true*; en caso contrario, devuelve *false*.
Number.isNaN(exp);[★]	Versión más robusta de *isNaN()*. Solo los valores de tipo numérico, que también son *NaN*, devuelven *true*, aunque *exp* no se pueda convertir.

[★] *EcmaScript 2015 o ES6 o JS ES6 o JavaScript 6*.

Ejemplos:

```
isNaN(NaN); //Devuelve true
isNaN("string"); //Devuelve true (cambia con Number.isNaN())
isNaN("12"); //Devuelve false
isNaN(12); //Devuelve false

Number.isNaN(NaN); // Devuelve true
Number.isNaN(Number.NaN); // Devuelve true
Number.isNaN(0 / 0); // Devuelve true


//Los siguientes hubieran devuelto true con isNaN()
Number.isNaN("NaN"); // Devuelve false
Number.isNaN(undefined); // Devuelve false
Number.isNaN({});// Devuelve false
Number.isNaN("blabla"); // Devuelve false

//Los siguientes devuelven false
Number.isNaN(true);
Number.isNaN(null);
Number.isNaN(37);
Number.isNaN("37");
Number.isNaN("37.37");
Number.isNaN("");
Number.isNaN(" ");
```

PROPIEDADES DE NUMBER

Propiedad	Descripción
`Number.MAX_VALUE`	Mayor número posible en JavaScript.
`Number.MIN_VALUE`	Menor número posible en JavaScript.
`Number.NEGATIVE_INFINITY`	`Infinity`
`Number.POSITIVE_INFINITY`	`-Infinity`
`Number.NaN`	`NaN`

Son propiedades estáticas del objeto `Number`: solo se puede utilizar `Number.MAX_VALUE`; `n.MAX_VALUE` devolvería `undefined`.

MÉTODOS PARA OBTENER UN NÚMERO COMO CADENA

Método	Devuelve una cadena
***numero*.toFixed([*num*]);**	Con el número de decimales que se pasa como parámetro. El valor por defecto es 0. El método redondea el valor, si es necesario.

`numero.toPrecision([num]);`	Con el número de cifras indicadas. El valor por defecto es 0. El método redondea el valor, si es necesario.
`numero.toExponential(num);`	En formato exponencial con el número de decimales indicado.

`numero.toString([num]);`	Si no se pasa el parámetro *num*: devolverá el número en base 10.
	Si se pasa un número, devolverá el número en esa base.
	Ej.
	`numero.toString(2);`
	`numero.toString(8);`
	`numero.toString(16);`

OTROS MÉTODOS

Método	Devuelve una cadena
`numero.valueOf();`	Valor primitivo de un número.

MÉTODOS GLOBALES PARA CONVERTIR VALORES A NÚMEROS

Método	Ejemplo	Devuelve
`Number();`	Number("10"); Number("10.3"); Number("103e-1"); Number("0x11"); Number("0b11"); Number("0o11"); Number(""); Number(null); Number(true); Number(false); Number(new Date()); Number("10a"); Number("a10");	10 10.3 10.3 17 3 9 0 0 1 0 Nº milisegundos NaN NaN

`parseInt(cadena);`	parseInt("10");
	parseInt("10a");
	parseInt("a10");
		10
		10
		NaN

`parseFloat(cadena);`	parseFloat("10.98");	10.98

Objetos nativos: Math

El objeto Math facilita propiedades y métodos para realizar operaciones matemáticas.
No tiene asociado constructor. **No es correcto:** `var x = new Math().`

PROPIEDADES

Propiedad	Descripción
***Math*.E**	Número e.
***Math*.LN2**	Logaritmo neperiano de 2.
***Math*.LN10**	Logaritmo neperiano de 10.
***Math*.LOG2E**	Logaritmo en base 2 del número e.
***Math*.LOG10E**	Logaritmo en base 10 del número e.
***Math*.PI**	Número PI.
***Math*.SQRT1_2**	Raíz cuadrada de ½.
***Math*.SQRT2**	Raíz cuadrada de 2.

MÉTODOS

Métodos	Devuelve
Math.random();	Un número aleatorio entre 0 y 1.
Math.max(v1, v2, …);	Máximo de una lista de valores.
Math.min(v1, v2, …);	Mínimo de una lista de valores.
Math.round(*num*);	*num* redondeado.
Math.ceil(*num*);	Función techo *num* redondeado a la alta.
Math.floor(*num*);	Función suelo *num* redondeado a la baja.
Math.pow(*base*, *expo*);	*base* elevada a *expo* (equivalente a *base****expo*).
Math.sqrt(*num*);	Raíz cuadrada de *num.*
Math.abs(*num*);	Valor absoluto de *num.*
Math.sin(*x*); **Math.cos(*x*);** **Math.tan(*x*);** **Math.asin(*x*);** **Math.acos(*x*);** **Math.atan(*x*);** **Math.atan2(y,*x*);**	Valores trigonométricos.

Ejemplo: Número aleatorio entre 0 y 10:

```
aleatorio = Math.floor(Math.random()*11);
```

Objetos nativos: Boolean

VALORES QUE SE EVALÚAN COMO VERDADEROS Y COMO FALSOS

En general, todo lo que tiene valor, se evalúa como *true*; lo que no tiene valor se evalúa como *false*. Una variable que se define, pero no se le da valor, se evalúa como *undefined*.

Se evalúan como verdaderos	Se evalúan como falsos
`true`	`false`
`5 < 6`	`5 > 6`
`100, -14, 3.4`	`0, -0`
`"hola", "false"` (*cadenas no vacías*)	`""` (*cadena vacía*)
	`undefined`
	`null`
	`NaN`

BOOLEANOS COMO TIPOS PRIMITIVOS Y COMO OBJETOS

Los booleanos pueden ser tipos primitivos

```
var x = false;
```

u objetos

```
var y = new Boolean(false);
```

Esta última forma no está aconsejada porque complica el código.

Por ejemplo:

`x == y` Es true.

`x === y` Es false porque al comparar dos objetos, devuelve false.

Objetos nativos: Date

Los objetos Date internamente guardan la fecha como el **número de milisegundos desde el 1 January 1970 00:00:00 UTC**[1]. Las fechas anteriores se almacenan con un número negativo.

Los **meses** se devuelven como un número. Comienzan en 0 (0 es enero, 1 febrero, …).

Los **días de la semana** se devuelven como un número (el 0 es domingo, 1 lunes, …).

CREAR OBJETOS DE TIPO DATE

Con la fecha y hora actual:

```
var actual = new Date();
```

A partir del número de milisegundos:

```
var fecha = new Date(numMilisegundos);
```

[1] UTC (Universal Time Coordinated) es lo mismo que GMT (Greenwich Mean Time).

Donde:

- *numMilisegundos*: es de tipo entero. Número de milisegundos desde el 1/1/1970.

Pasando como parámetros: año, mes, día (opcional: hora, min, seg, miliseg.):

```
var fecha = new Date(anno, mes, [dia , hora, min, seg, miliseg]);
```

Donde:

- Todos los parámetros son de tipo entero.
- Los valores para día, hora, minutos, segundos y milisegundos, son opcionales. Si no se indican, toman el valor 0.

Creación de un objeto Date a partir de una cadena:

```
var fecha = new Date(cadenaFecha);
```

Donde

- *cadenaFecha*: es un String, que admite los siguientes formatos:
  - `"Wed Mar 25 2015 09:23:45 GMT +0100 (hora estándar de Europa central)"`
  - `"October 12, 2016 10:30:00"`
  - `"January 25 2015"`
  - `"Jan 25 2015"`
  - `"25 Jan 2015"`
  - `"2016-05-12T12:34:25Z"` (la T separa la fecha de la hora y la Z indica la zona horaria)
  - `"2015-03-25T12:00:00-06:30"` (para modificar la zona horaria, lugar de Z pondremos +HH:MM o -HH:MM)
  - `"2016-05-12"` (DD/MM/YYYY no está definido, algunos navegadores averiguarán el resultado esperado, otros pueden devolver NaN)
  - `"2016-05"`
  - `"2016"`
  - `"05/12/2016"` (YYYY/MM/DD no está definido, algunos navegadores averiguarán el resultado esperado, otros pueden devolver NaN)

MÉTODOS GET: Obtener día, mes, ...

Método	Devuelve un número entero
***fecha*.getDay();**	Día de la semana: 0 = domingo, 1 = lunes,…
***fecha*.getDate();**	Día del mes: 1, 2, …
***fecha*.getMonth();**	Mes: 0 = enero, 1 = febrero, …
***fecha*.getFullYear();**	Año (cuatro dígitos).
***fecha*.getHours();**	Hora.
***fecha*.getMinutes();**	Minutos.

`fecha.getSeconds();`	Segundos.
`fecha.getMilliseconds();`	Milisegundos.
`fecha.getTime();` `Date.now();`	Número de milisegundos desde el 1/1/1970.

MÉTODOS SET: Establecer día, mes…

Método	Establece
`fecha.setDate(dia);`	*dia*: 1, 2, … Día del mes.
`fecha.setMonth(mes);`	*mes*: 0 = enero, 1 = febrero, …
`fecha.setFullYear(año);`	*año*: (cuatro dígitos).
`fecha.setHours(hora);`	*hora*: 0-24.
`fecha.setMinutes(min);`	*min*: 0-59.
`fecha.setSeconds(seg);`	*seg*: 0-59.
`fecha.setMilliseconds(ms);`	*ms*: milisegundos.

MÉTODOS PARA OBTENER LA FECHA COMO CADENA

Método	Devuelve una cadena similar a esta
`fecha.toString();`	Sun Nov 18 2018 18:28:27 GMT+0100 (hora estándar de Europa central)
`fecha.toUTCString();`	Sun, 18 Nov 2018 17:28:53 GMT
`fecha.toDateString();`	Sun Nov 18 2018

MÉTODO PARA OBTENER UNA FECHA EN MILISEGUNDOS

Método	Devuelve un número entero
`Date.parse(cadenaFecha);`	Número de milisegundos entre *cadenaFecha* y January 1, 1970.

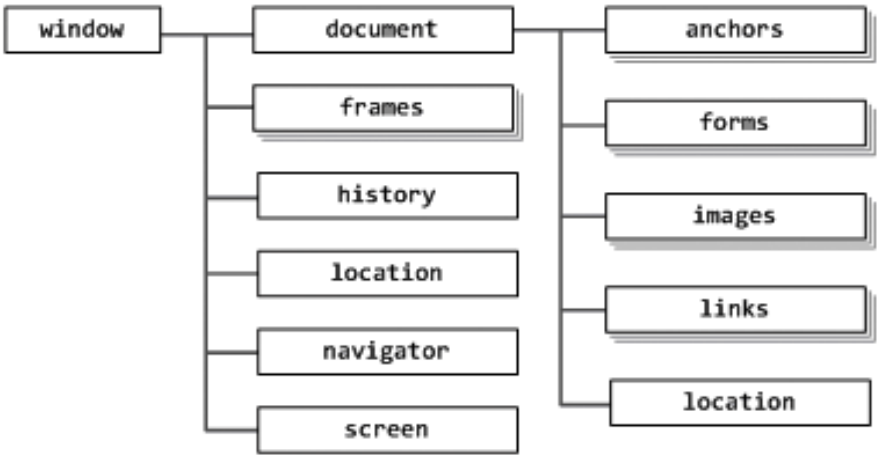
Objetos asociados al navegador: introducción

El modelo de objetos del navegador o BOM (Browser Object Model) permite a JavaScript tener una comunicación con el navegador de internet, ya que nos permite acceder y modificar las propiedades de sus ventanas.

Gracias al BOM, podemos redimensionar o mover la ventana del navegador, modificar el texto de la barra de estado y manipular opciones no relacionadas con el contenido propiamente dicho.

Tiene la desventaja de que no existe ningún estándar que defina los mínimos de interoperabilidad entre los navegadores.

El BOM está compuesto por varios objetos relacionados entre sí:



Objetos asociados al navegador: window

Representa una ventana abierta en un navegador.
La ventana actual es un objeto **window** del tipo **Window**.

Todos los objetos, funciones y variables que utilizamos en nuestros scripts, pasan a ser automáticamente miembros del objeto **window**:

- La variables globales y el objeto **document** del DOM: son propiedades
- Las funciones: son métodos.

Notas:

- Las propiedades/métodos del objeto Window pueden variar en cada navegador.
- Si la ventana tiene etiquetas tipo <iframe>, el navegador crea un objeto Window para el HTML inicial y otro para cada <iframe>.

Ventanas

PROPIEDADES

Propiedad	Descripción
***miVentana*.closed**	Boolean. Indica si la ventana está abierta o cerrada.
***miVentana*.innerHeight**	Altura (en píxeles) de la ventana del navegador incluyendo, si existe, la barra de desplazamiento horizontal.
***miVentana*.innerWidth**	Anchura (en píxeles) de la ventana del navegador incluyendo, si existe, la barra de desplazamiento vertical.
***miVentana*.name**	Establece/obtiene el nombre de la ventana.
***miVentana*.outerHeight**	Altura (en píxeles) de la ventana del navegador, incluyendo la parte superior y los bordes para cambiar de tamaño.
***miVentana*.outerWidth**	Ancho (en píxeles) de la ventana del navegador, incluyendo la barra lateral (si hay), la parte superior y los bordes para cambiar de tamaño.
***miVentana*.pageXOffset**	Alias de **scrollX**.
***miVentana*.pageYOffset**	Alias de **scrollY**.
***miVentana*.screenX**	Distancia horizontal, en píxeles CSS, desde el borde izquierdo del navegador hasta el lado izquierdo de la pantalla.
***miVentana*.screenY**	Distancia horizontal, en píxeles CSS, desde el borde superior del navegador hasta el lado superior de la pantalla.
***miVentana*.scrollX**	Número de píxeles que el documento está haciendo scroll horizontalmente.
***miVentana*.scrollY**	Número de píxeles que el documento está haciendo scroll verticalmente.

MÉTODOS

Método	Descripción
window.open([*url*], [*nombreVentana*], [*opciones*], [*historial*]);	Carga la URL en el contexto del navegador (window, iframe o tab) con el nombre especificado. Si el nombre no existe, se abre una nueva ventana y la URL se carga en su contexto. Donde: • url: es opcional; si no se especifica, se carga una página en blanco.

	• nombreVentana: puede ser:
	  ○ "_blank": nueva ventana o pestaña (valor por defecto)
	  ○ "_parent" frame padre
	  ○ "_self": reemplaza la página actual
	  ○ "top": reemplaza todos los frames
	  ○ "nombre": que le demos nosotros para poder usar la ventana como destino de enlaces o formularios.
	• opciones: lista de opciones separadas por comas para definir la ventana.
	• historial: puede ser true (se reemplaza la entrada del historial) o false (se añade al historial)

***miVentana*.focus();**	Pasa el foco a la ventana.
***miVentana*.print();**	Abre el cuadro de diálogo Imprimir del documento actual.
***miVentana*.stop();**	Detiene la carga de la ventana.
***miVentana*.close();**	Cierra la ventana.
***miVentana*.resizeBy(*xIncr, yIncr*);**	(*) Aumenta/disminuye el tamaño de la ventana en los píxeles indicados como parámetros.
***miVentana*.resizeTo(*xValor, yValor*);**	(*) Establece el tamaño de la ventana a los píxeles indicados como parámetros (***xValor*** corresponde a **outerWidth**; e ***yValor*** a **outerHeight**).
***miVentana*.moveBy(*xIncr, yIncr*);**	(*) Mueve la ventana los píxeles indicados como parámetros.
***miVentana*.moveTo(*xValor, yValor*);**	(*) Mueve la ventana a las coordenadas especificadas.
***miVentana*.scrollBy(*xIncr, yIncr*);**	Hace scroll en la ventana, en los píxeles indicados como parámetros.
***miVentana*.scrollTo(*xValor, yValor*);**	Hace scroll en la ventana, hasta las coordenadas indicadas como parámetro.

(*) Por razones de seguridad, solo se puede cambiar el tamaño y/o mover ventanas creadas con **window.open()**, y que solo contengan una pestaña.

Ejemplos:

```
v = window.open("https://www.google.com");
x = window.open("","width=500, height=200");
x.document.write("<h1>Mi ventana</h1>");
```

Cuadros de diálogo

MÉTODOS

Método	Descripción
`.alert("mensaje");`	Muestra el mensaje. No devuelve nada.
`.prompt("mensaje" [, <texto por defecto>]);`	Devuelve una cadena con el texto introducido por el usuario, o **null**.
`.confirm("mensaje");`	Devuelve `true` o `false`. Si el usuario hace clic en Cancelar o cierra la ventana, devuelve `false`.

Instrucciones de tiempo

FUNCION	Descripción
`clearInterval(vbleTimeOut);`	Anula la ejecución programada con un **`setTimeOut()`** o **`setInterval()`**.
`clearTimeOut(vbleTimeOut);`	Anula la ejecución programada con un **`setTimeOut()`** o **`setInterval()`**.
`setInterval(función, milisegundos);`	Repite una función cada número de milisegundos indicado. Devuelve un número distinto de cero, que identifica a esta instrucción.
`setTimeOut(function, milisegundos, par1, par2…);`	Es un contador que ejecuta una función una vez que el tiempo indicado haya expirado. Devuelve un número distinto de cero, que identifica al temporizador que fue creado al llamar a esta función. Par1, par2, … son parámetros adicionales que se le pueden pasar a la función.

`setTimeOut()` y **`setInterval()`** comparten una pila de IDs, por lo que **`clearTimeOut()`** y **`clearInterval()`** pueden ejecutarse indistintamente. Por motivos de claridad y para facilitar el mantenimiento, es importante utilizarlos como corresponde.

Ejemplo: Cuando se pulsa el botón, espera 3000 milisegundos y ejecuta una función.

```
<button onclick="setTimeOut(f, 3000)">Pulsa</button>
<script>
function f() {
}
</script>
```

Ejemplo: Como el anterior, pero además creamos otro botón que, si se pulsa después del primero, pero antes de que pasen los 3000 ms, anula la orden de ejecutar la función.

```
<button onclick="v=setTimeOut(f, 3000)">Pulsa</button>
<button onclick="clearTimeOut(v)">Cancela</button>
<script>
function f() {
}
</script>
```

Ejemplo: Cuando se pulsa el primer botón, ordena que se repita la ejecución de una función cada 1000 ms.
Si se pulsa el segundo botón, anula la orden de repetir la ejecución de la función.

```
<button onclick="v=setInterval(f, 1000)">Pulsa</button>
<button onclick="clearInterval(v)">Cancela</button>
<script>
function f() {
     var fecha = new Date();
     document.getElementById("r").innerHTML=fecha.getSeconds();
}
</script>
```

Objetos asociados al navegador: navigator

Nos va a permitir obtener información sobre el propio navegador. Un objeto **`Navigator`** puede ser accedido utilizando la propiedad de solo lectura **`window.navigator`**.

Propiedad	Descripción
`.appCodeName`	(*) Devuelve siempre “Mozilla”, en cualquier navegador.
`.appName`	(*) Devuelve siempre “Netscape”, en cualquier navegador.
`.appVersion`	(*) Devuelve “4.0” o una cadena representando información de la versión del navegador.
`.cookieEnabled`	Devuelve una propiedad que indica si las cookies están activadas o no.
`.platform`	Devuelve una cadena que representa la plataforma (sistema operativo) sobre el que se está ejecutando el navegador.
`.product`	(*) Devuelve siempre “Gecko”, en cualquier navegador.
`.userAgent`	Devuelve una cadena con información sobre el agente de usuario (navegador).
`.language`	Devuelve el idioma en el que está configurado el navegador.
`.online`	Devuelve *true* si el navegador está en línea.

(*) No confíes en esta función para obtener un valor significativo.

Métodos	Descripción
`.javaEnabled();`	Devuelve un valor boolean que indica si el navegador está habilitado en ese momento para soportar la ejecución de programas escritos en Java.

Objetos asociados al navegador: screen

Se utiliza para obtener información sobre la pantalla del usuario, como puede ser, por ejemplo, la resolución del monitor. Se puede acceder utilizando **`window.screen.`**

Los navegadores determinan sobre qué pantalla informar, detectando qué pantalla tiene el centro de la ventana del navegador.

Propiedad	Descripción
`.availHeight`	Altura disponible en la pantalla para el uso de las ventanas.
`.availWidth`	Anchura disponible en la pantalla para el uso de las ventanas.
`.height`	Altura total de la pantalla.
`.width`	Anchura total de la pantalla.
`.colorDepth`	(*) Profundidad del color de la pantalla.
`.pixelDepth`	(*) Profundidad de la pantalla en píxeles.

(*) En los ordenadores actuales, este valor es el mismo.

Objetos asociados al navegador: history

Nos permite manipular el historial de la sesión, es decir, las páginas visitadas en esa pestaña o en el marco donde está cargada la página.

Se puede acceder utilizando la propiedad de solo lectura **`window.history.`**

Propiedad	Descripción
`.length`	Número de páginas en el historial, incluyendo la página cargada.

Método	Descripción
`.back();`	Va a la página anterior en el historial. Equivale a hacer clic en el botón de volver atrás del navegador.

	Equivale a **`window.history.go(-1)`**
`.forward();`	Va a la página siguiente en el historial. Equivale a hacer clic en el botón de ir adelante del navegador. Equivale a **`window.history.go(1)`**
`.go(incrPag);`	Carga una página del histórico de la sesión, a través de un índice que toma como referencia la página actual. Un valor positivo permite avanzar. Un valor negativo permite retroceder. Si el índice excede los márgenes, no se realiza ninguna acción.

Objetos asociados al navegador: location

Window.location representa la URL de la página HTML que se muestra en la ventana del navegador, lo que nos proporciona varias propiedades muy útiles.

Propiedad	Descripción
`.host`	Nombre del servidor seguido del puerto.
`.hostname`	Nombre del servidor o dominio (ej. www.google.es).
`.href`	URL completa de la página actual.
`.pathname`	Directorio y subdirectorio junto con el nombre del recurso de la URL (lo que se encuentra después del host). Ejemplo: /index.html
`.port`	Puerto especificado en la URL, si lo hay.
`.protocol`	Protocolo utilizado en la petición (lo anterior a las //).
`.search`	Parte correspondiente a la query. Ej: ?id=10&ciclo=CFGS
`.hash`	Anclaje de la URL, lo que va detrás del símbolo #.

Método	Descripción
`.assign(url);`	Sustituye el documento actual por el indicado en la url, pero mantiene el actual en el historial del navegador.
`.reload();`	Vuelve a cargar el documento actual en el navegador.
`.replace(url);`	Sustituye el documento actual por el indicado en la url, pero borrando el actual del historial del navegador (como si nunca se hubiera visitado).