08 OBJETOS DEL NAVEGADOR (BOM)

OBJETOS DEL NAVEGADOR. Browser Object Model o BOM

http://librosweb.es/libro/ajax/capitulo_5.html

http://www.w3schools.com/js/js_window.asp

https://developer.mozilla.org/es/docs/Web/API/Window

Las versiones 3.0 de los navegadores Internet Explorer y Netscape Navigator introdujeron el concepto de **Browser Object Model o BOM**, que permite acceder y modificar las propiedades de las ventanas del propio navegador.

Mediante BOM es posible, entre otras cosas:

- redimensionar y mover la ventana del navegador
- crear y cerrar ventanas
- modificar el texto que se muestra en la barra de estado
- obtener información sobre el propio navegador
- trabajar con propiedades de la página actual y de la pantalla de usuario
- controlar el tiempo de ejecución de instrucciones

El mayor inconveniente de BOM:

> Ninguna entidad se encarga de estandarizarlo o definir unos mínimos de interoperabilidad entre navegadores.

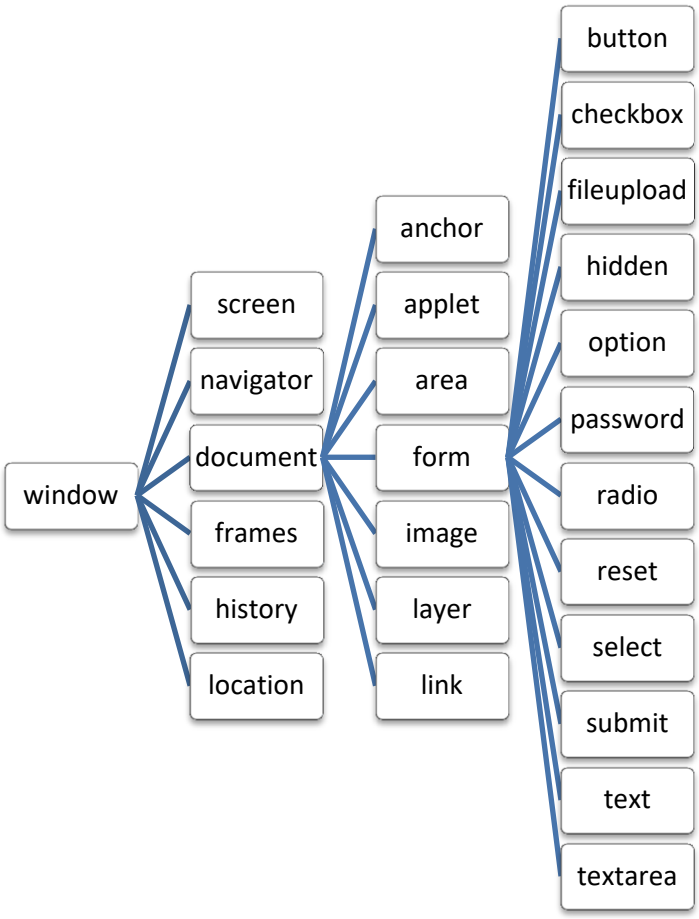
Existen una serie de **objetos predefinidos de JavaScript** (ventana, documento, ubicación, formulario, marco) que se crean automáticamente cuando se carga una página en el navegador y que podemos manipular mediante JavaScript para cambiar sus propiedades, realizar tareas a través de sus métodos o ejecutar un evento relacionado con el mismo objeto.

Los objetos de JavaScript se ordenan de modo jerárquico.

En lo más alto de la jerarquía se encuentra el objeto *Window*, que representa la ventana del navegador. De entre los distintos objetos generados automáticamente destacan los siguientes:

- window: es el objeto principal del cual "cuelgan" los demás como si fueran propiedades suyas. Se crea un objeto de este tipo para cada ventana que pueda ser lanzada desde una página Web.
- **navigator**: contiene información acerca del navegador, tal como nombre, versión, tipos MIME soportados por el cliente...
- **screen**: contiene información acerca de la pantalla
- **location**: con información acerca del URL que estemos visualizando.
- **history**: historial en el que se guardan los URL ya visitados.
- **document**: es el contenedor de todos los objetos que se hayan insertado en la página web: enlaces, formularios, imágenes o marcos.
- **frames**: Matriz conteniendo los distintos objetos frame que puede contener una ventana. Cada frame es a su vez otra ventana.

Jerarquía de los objetos de JavaScript



Las propiedades del objeto document van en función de los objetos que hayamos insertado en la página. Cada objeto que incluyamos en la página será una propiedad del objeto document.

Cada vez que se desea acceder a una propiedad o un método de un objeto, hemos de detallar cuáles son sus padres mediante el uso de puntos. Así, la propiedad value de un botón de un formulario se llamaría en realidad

window.document.form.button.value

Aunque el padre, window, se puede quitar si es la misma ventana en la que está el formulario.

Hemos de tener en cuenta, que el navegador va creando los objetos en el momento que los lee, y que la página la lee desde arriba hacia abajo. Por ello, no debemos usar objetos antes de su creación. Solo al definir funciones en la cabecera podemos romper esta regla, ya que la definición de las funciones no implica su uso en el momento que se leen, sino que se llaman después desde el cuerpo de la página. Por tanto, en la definición de las funciones sí podremos hacer referencia a objetos aún no creados (que son todos los de document) aunque no podremos usar tales funciones hasta que dichos objetos no estén creados.

1. Objeto window

El objeto window es el más importante. A partir de él se pueden crear nuevos objetos window que serán nuevas ventanas ejecutando el navegador. Tales ventanas se pueden controlar desde la ventana padre. Además permite cerrar ventanas, actúa sobre los marcos, puede sacar mensajes (de error u otro tipo), confirmación y entrada de datos, y como se ha dicho, mantiene una matriz por cada tipo de elemento que puede haber en un documento, formularios, enlaces, imágenes y marcos.

Apertura de una ventana

(Ejemplo "01 abrir y cerrar.html")

El método **open()** abre una nueva ventana.

Es necesario asignar la nueva ventana a una variable si posteriormente queremos operar con ella (por ejemplo para escribir en ella o cerrarla)

```
nuevaVentana = window.open ("dirección URL");
```

En realidad, son varios los parámetros que se pasan a una ventana al crearla:

```
variable=window.open ("dirección URL", "nombre de la ventana", "parámetros de apertura" );
```

window.open(URL,nombre,especificaciones)

Aquí puedes ver todos los parámetros disponibles:

https://www.w3schools.com/jsref/met_win_open.asp

Los más utilizados los vemos a continuación:

<table>
<tr><th>Parámetro</th><th>Descripción</th></tr>
<tr><td>URL</td><td>Opcional. Especifica la dirección URL de la página que se abrirá. Si no se especifica una URL, se abre una nueva ventana con about: blank</td></tr>
<tr><td rowspan="4">nombre</td><td>Opcional. Especifica el atributo de destino o el nombre de la ventana. Los valores siguientes son compatibles:</td></tr>
<tr><td>_blank - URL se carga en una ventana nueva.</td></tr>
<tr><td>_self - URL reemplaza la página actual</td></tr>
<tr><td>Nombre - El nombre de la ventana. Este es el nombre que se utilizará posteriormente en los target de los enlaces para indicar que el enlace se abra en esa ventana</td></tr>
<tr><td>especificaciones</td><td>Opcional. Una lista separada por comas de los elementos, sin dejar espacios en blanco. Los valores siguientes son compatibles:
<table>
<tr><td>height= pixeles</td><td>La altura de la ventana. Min. valor es 100</td></tr>
<tr><td>left = píxeles</td><td>La posición izquierda de la ventana</td></tr>
<tr><td>top = píxeles</td><td>La posición superior de la ventana.</td></tr>
<tr><td>width = píxeles</td><td>La anchura de la ventana. Min. valor es 100</td></tr>
</table>
</td></tr>
</table>

Cerrar una ventana

(Ejemplo "01 abrir y cerrar.html")

El método close() cierra una ventana.

Para cerrar una ventana se utiliza el método close() de las mismas:

```
variable_de_ventana.close()
```

window.close() si es la propia ventana

Importante: Los scripts no pueden cerrar ventanas que no hayan sido abiertas por un script.

Mensajes de alerta, confirmación y entrada de datos.

(Ejemplo "02 alert confirm prompt.html")

El método **alert**(*mensaje*) nos permite mostrar un mensaje de alerta

El método **confirm**(*mensaje*) muestra un cuadro de diálogo con un mensaje y los botones Aceptar y Cancelar. Devuelve true si se pulsa aceptar y false si se pulsa cancelar.

El método **prompt** (*mensaje,valor por defecto)* permite recibir datos por pantalla

Mover y redimensionar una ventana.

(Ejemplo "03 mover y redimensionar.html")

Métodos resizeBy, resizeTo, moveBy, moveTo

Temporizadores

(Ejemplo "04 temporizadores.html")

setInterval(codigo, milisegundos)

Parámetro	Descripción
código	Obligatorio. La función que se ejecutará
milisegundos	Obligatorio. Los intervalos de tiempo (en milisegundos) sobre la frecuencia para ejecutar el código

El método setInterval llama a una función o evalúa una expresión a intervalos especificados (en milisegundos)

Tiene por parámetros obligatorios el código de la función que se ejecutará y la cantidad de milisegundos del intervalo de ejecución

Se suele utilizar de la siguiente manera:

variable= window.setInterval (function(){código de la función;}, milisegundos);

Para parar la ejecución hacemos uso de "variable"

clearInterval(variable);

```
function saludar(){
        a=window.setInterval(function(){ alert("Hola");}, 3000);
 }
function parar(){
        clearInterval(a);
}
```

setTimeout(codigo, milisegundos)

El método setTimeout ejecuta una función tras un número especificado de milisegundos

Parámetro	Descripción
código	Obligatorio. La función que se ejecutará
milisegundos	Obligatorio. El número de milisegundos de espera antes de ejecutar el código

```
function saludar(){
        window.setTimeoutl(function(){ alert("Hola");}, 3000);
 }
```

Ver ejemplos:

04 temporizadores.html

http://www.w3schools.com/js/tryit.asp?filename=tryjs_setinterval

2. Objeto Navigator

http://www.w3schools.com/js/js_window_navigator.asp

(Ver)

Podemos referirnos a él a través del objeto window: window.navigator o sin el prefijo window

Contiene información sobre el navegador, permite identificar las características de la plataforma sobre la que se está ejecutando nuestra página.

Ejemplos de algunas propiedades:

Propiedad	Descripción
appCodeName	Devuelve el nombre del código del navegador Realmente devuelve el nombre del grupo de trabajo que desarrolló el código de los primeros navegadores, nombre con el que Netscape a su primer explorador. Tanto Microsoft Internet Explorer como Netscape devuelven el valor Mozilla. No es una propiedad demasiado útil, tan sólo nos permite detectar exploradores basados en algún código poco habitual.
appName	Devuelve el nombre del navegador
cookieEnabled	Determina si las cookies están habilitadas o no
userAgent	Devuelve una información completa sobre el agente de usuario(normalmente el navegador, pero podría ser cualquier agente que envía una petición HTTP)

3. Objeto Screen

http://www.w3schools.com/js/js_window_screen.asp

Podemos referirnos a él a través del objeto window: window.screen o sin el prefijo window

Ofrece información sobre la pantalla, el monitor, utilizada por el usuario.

Cuenta con seis propiedades que son solamente de lectura y no posee ningún método. Las más destacadas:

Propiedad	Descripción
availHeight	Devuelve la altura de la pantalla para el uso de ventanas (excepto la barra de tareas de Windows)
availWidth	Devuelve el ancho de la pantalla

Puede resultar útil saber la resolución de la pantalla de usuario para que el diseñador web adapte sus diseños antes de cargarlos.

4. Objeto history

http://www.w3schools.com/js/js_window_history.asp

Podemos referirnos a él a través del objeto window: window.history o sin el prefijo window

Desde JavaScript es posible moverse por el historial de navegación con los métodos del objeto history.

Almacena las referencias de todos los sitios web visitados. Se utiliza para movernos hacia delante y hacia atrás en esta lista. Al ser una lista de referencias, no podemos acceder a los nombres de las direcciones URL visitadas, ya que son información privada del usuario.

Método	Descripción
back()	Carga la URL anterior en la lista del historial.
forward()	Carga la URL siguiente en la lista del historial.
go()	Recibe como parámetro un número entero positivo o negativo y navega hacia delante o hacia atrás respectivamente ese número de elementos del historial.

Propiedad	Descripción
length	Corresponde al número de páginas que han sido visitadas

La razón por la que este objeto tiene una funcionalidad tan limitada es por temas de seguridad y privacidad. Si se permitiese conocer mediante el uso de scripts cualquier detalle sobre los sitios visitados, se estaría facilitando una información de la que una empresa podría obtener valiosos datos para hacer un estudio de mercado.

En aras de la seguridad no está permitido el acceso mediante script a ninguna de las características de una página residente en otro marco o ventana si no pertenece al mismo dominio de la página donde se encuentra el script.

```
<html>
<head>
</head>
<body>
<input type="button" value="Atrás"
       onclick="history.back()"> <br>
<input type="button" value="Adelante"
       onclick="history.forward()"> <br>
<input type="button" value="2 hacia atrás"
       onclick="history.go(-2)"> <br>
<input type="button" value="Sitios visitados"
       onclick="alert(history.length)"> <br>
</body>
</html>
```

5. Objeto location

http://www.w3schools.com/js/js_window_location.asp

Podemos referirnos a él a través del objeto window: window.location o sin el prefijo window

Contiene la información referente a la localización (origen) del documento mostrado en una determinada ventana.

Se utiliza principalmente para redireccionar el navegador a una nueva página

Su propiedad más importante es ***href:***

Contiene la URL completa del documento en forma de cadena de texto.

Es una propiedad de lectura y escritura, por lo que se puede averiguar la localización de una página con ella y se puede obligar a la ventana a mostrar otra página diferente.

Es la propiedad predeterminada del objeto, por lo que ni siquiera es necesario especificarla para acceder a su contenido

location.href = http://www.google.es/

location = http://www.google.es/

El resto de propiedades existen básicamente como medio directo de obtención de diversas partes de href

Método	Descripción
assign()	Carga un nuevo documento
reload()	Vuelve a cargar el documento actual
replace()	Reemplaza el documento actual por uno nuevo. Además de sustituirlo en la ventana actual, sustituye su entrada en el historial de páginas visitadas. La página que estuviera anteriormente ya no puede ser accedida a través de los botones de navegación, puesto que es sustituida su entrada.

```
<html>
<head>
<script  type="text/javascript">
document.write ('<br>href '+location.href+'<br>');
//location="http://www.google.es/";
</script>
</head>
<body>
<input type="button" value="Actualizar"
                     onclick="location.reload()"> <br>
<input type="button" value="Cambio a google"
                     onclick="location='http://www.google.es/';"> <br>
<input type="button" value="Reemplazo por El País"
                     onclick="location.replace('http://www.elpais.es/')"> <br>
<input type="button" value="Asigno Marca"
                     onclick="location.assign('http://www.marca.com/')"> <br>
</body>
</html>
```

6. Objeto document

Representa el propio documento que se está mostrando en la ventana del navegador.

Se utiliza para acceder a sus propiedades, así como a los elementos que lo constituyen.

Cuenta con una serie de subobjetos como son los vínculos, puntos de anclaje, imágenes o formularios, que representan el verdadero potencial del objeto document.

Algunas de sus propiedades:

Propiedad	Descripción
alinkColor	Corresponde al color de los vínculos activos de la página

anchors	Devuelve un array con todos los anclajes (etiquetas <a name>) del documento document.anchors.length nos dice cuantos document.anchors[i].name el anclaje
bgColor	Corresponde al color del fondo del documento
cookie	Devuelve todos los pares nombre / valor de las cookies en el documento
forms	Se trata de un array con todos los formularios del documento. Los formularios tienen a su vez elementos (cajas de texto, botones, etc) que tienen sus propias propiedades y métodos, document.forms.length nos dice cuantos document.forms[i].name sus nombres
images	Devuelve un array de todas las imágenes en el documento document.images.length nos dice cuantas document.images[i].src los archivos con sus rutas
lastModified	Devuelve la fecha y la hora que se modificó el documento última
linkColor	Corresponde al color de los vínculos de la página
links	Devuelve un array de todos los enlaces (etiqueta <a href>) del documento document.links.length nos dice cuantos document.links[i] los enlaces
title	Establece o devuelve el título del documento
URL	Devuelve el URL completo del documento
vlinkColor	Corresponde al color de los vínculos visitados de la página

Método	**Descripción**
close()	Cierra el flujo de salida abierto previamente con document.open ()
open()	Se abre un flujo de salida para recoger la salida de document.write () o document.writeln ()

write()	Escribe expresiones de código HTML o JavaScript en un documento
writeln()	Igual que escribir (), pero añade un carácter de nueva línea después de cada declaración. Sólo tiene efecto visible en la página si se utilizan las etiquetas <pre> </pre> para texto con formato, pues en HTML un retorno de carro se expresa con

Se pueden utilizar los métodos open() y close() para definir bloques de escritura de texto al documento. El contenido de un bloque no se envía al documento para su visualización hasta que se cierra.

Al hacer open(), si el documento ya existía, se borrará.