3. Los objetos del BOM

En este apartado, se van a presentar los objetos junto con sus principales propiedades y métodos que forman parte de BOM, de manera que en el siguiente apartado se presentará el DOM, es decir, todo el modelo de objetos que desciende desde el objeto Document.

Web recomendada

Para información más detallada y actualizada de las propiedades y los métodos de los objetos del BOM, podéis visitar la página de w3schools.

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

Como es de esperar, la lista de objetos presentados es dinámica y evoluciona en paralelo a la aparición de nuevas versiones de los navegadores. La utopía de la estandarización sigue siendo un objetivo pendiente y no una realidad como sería deseable.

Es importante destacar que esta parte de BOM sufre el mayor número de divergencias, ya que este está directamente relacionado con el navegador.

Por tanto, el objetivo del apartado es la presentación de los objetos y sus componentes que, junto con algunos ejemplos, ayudarán a comprender el potencial del proceso de manipulación del DOM desde JavaScript. En ningún momento se plantea que se aprendan "de memoria" todos los objetos con sus métodos y propiedades, no tiene ningún sentido, siempre se pueden consultar en el propio apartado o en línea en internet.

En cuanto a esto último, y de cara a solucionar problemas con la compatibilidad del código, las siguientes referencias web muestran el modelo de objetos en cada uno de los navegadores actuales, incluyendo tanto el BOM como el DOM:

- Especificación del DOM de los navegadores basado en Mozilla: https://developer.mozilla.org/en/Gecko_DOM_Reference

- Especificación del DOM de los navegadores Internet Explorer: http://msdn.microsoft.com/en-us/default.aspx

3.1. El objeto Window

El objeto Window es el que se encuentra en el nivel más alto en la jerarquía de objetos y, por lo tanto, es el contenedor de todo aquello que se puede ver en el navegador.

Tan pronto como el navegador se ejecuta, y aunque este no visualice ningún documento, se crea una instancia de Window que es accesible desde JavaScript.

Pueden referenciarse todas las ventanas que están abiertas en la pantalla y pueden ser manipuladas por una instancia del objeto (siempre que hayan sido creadas desde la propia instancia que las intenta manipular). Tal como se verá en este subapartado, las propiedades de una instancia de Window incluyen su tamaño, componentes, posición, etc. Por su parte, los métodos van a permitir la creación y destrucción de ventanas genéricas, de ventanas de alerta o de confirmación, etc.

Debe tenerse en cuenta que el objeto Window puede representar la ventana del navegador o a un *frame* (cuando existen varios *frames* cada uno de ellos se considera un objeto Window independiente).

Así pues, un objeto Window se puede crear mediante alguna de las siguientes opciones:

- las etiquetas “body” o “frameset” de HTML, y
- el método open de JavaScript.

Por su parte, el acceso a las propiedades y los métodos del objeto Window siguen la sintaxis que se ha presentado en los apartados anteriores:

```
window.nombre_propiedad;
window.nombre_metodo([parámetros]);
```

Cuando se hace referencia a la ventana que contiene el propio documento, puede usarse la palabra *self* en sustitución de la palabra *window*. Por ejemplo, para cerrar la ventana actual se pueden usar indistintamente las formas window.close() y self.close().

De todos modos, con los métodos open() y close() es necesario usar las formas window.open() y window.close(), ya que aquellos se podrían confundir con los métodos del objeto Document (document.open() y document.close()).

La palabra *self* puede omitirse en los casos más simples y reservarse para aquellos casos que impliquen varios *frames*. Esto es debido a que, cuando se visualiza un documento, se asume el hecho de que ya existe una ventana, la que lo contiene. Por lo tanto, se puede acceder a las propiedades y los métodos de la ventana actual sin especificarla:

```
nombre_propiedad
nombre_metodo([parametros])
```

A continuación, se va a mostrar una tabla con las principales propiedades disponibles del objeto Window:

Tabla 8. Propiedades del objeto Window

Nombre	Descripción	Ejemplo

closed	Es un valor booleano que indica si una ventana ha sido cerrada. Cuando se cierra una ventana, el objeto Window continúa existiendo, pero con el valor de esta propiedad true.
	Es de solo lectura.
		`ventana = window.open( 'doc.htm','ventana1', 'width=400, height=100' )`
		`...`
		`if (ventana.closed)`
		`  alert("cerrada")`
		`else`
		`  alert("abierta")`

defaultStatus	Es el mensaje por defecto que se visualiza en la barra de estado del navegador.	**window.defaultStatus**="Página índice"
document	Hace referencia al objeto Document contenido en la ventana.	
frames	Es un *array* de objetos *frame*. Los *frames* son los generados por la etiqueta Frame del FRAMESET contenido en la página, y en el *array* están en el orden en el que se han creado en el código HTML. Es de solo lectura.	En una página que contenga dos *frames*, se puede acceder a estos de la siguiente manera: parent.frames["findice"] parent.frames["fcontenido"] o bien parent.frames[0] parent.frames[1]
history	Hace referencia al objeto History contenido en la ventana.	

length	Contiene el número de *frames* de la ventana.
	Es de solo lectura.
		`numframes = window.length;`
		`if (numframes == 0)`
		`  alert("no contiene frames");`
		`else`
		`  alert("contiene "+numframes+" frames");`

location	Hace referencia al objeto location contenido en la ventana.	
name	Cadena de texto que especifica el nombre de la ventana o marco. Es de solo lectura.	ventana = window.open('doc.htm','ventana1','width=400, height=100') alert(ventana.name)
navigator	Hace referencia al objeto Navigator contenido en la ventana.	
opener	Hace referencia al objeto Window que ha abierto la ventana actual. Esta propiedad persiste aunque el documento que ha hecho la llamada se haya descargado.	Cierra la ventana que ha abierto la ventana actual: window.opener.close() Cierra la ventana del navegador: top.opener.close()
parent	Hace referencia al objeto Window o Frame padre del marco actual. Es la ventana que contiene el *frameset*.	En una página que contenga dos *frames*, se puede acceder al segundo desde el primero con la siguiente sentencia: **parent**.frames[1]
screen	Hace referencia al objeto Screen del navegador.	
self	Es sinónimo de la ventana actual.	

status	Especifica el texto que se visualiza en la barra de estado del navegador.	`function ini_status() {`
		`  window.status = "Pulse en este link para acceder al índice"`
		`}`
		`<a href="indice.htm" onMouseOver = "ini_status()">Índice</a>`

Nombre	Descripción	Ejemplo
top	Hace referencia al objeto Window superior en la jerarquía de objetos del documento. Se utiliza para acceder al objeto Window contenedor desde un marco.	top.close()
window	Es la ventana o el *frame* actual.	

La siguiente tabla muestra los principales métodos del objeto Window. Al igual que en el caso de las propiedades, no se trata de una listado exhaustivo, sino que se presentan aquellos métodos más utilizados o considerados interesantes por el autor.

Tabla 9. Métodos del objeto Window

Nombre	Descripción	Sintaxis	Parámetros	Retorno
alert	Abre una caja de diálogo con un mensaje y el botón *Aceptar*.	alert(cadena de texto)	String (cadena de texto)	
blur	Elimina el foco de la ventana o *frame* especificados.	blur()		
clearInterval	Cancela el *timeout* que ha sido inicializado con el método setInterval.	clearInterval(intervalID)	intervalID se inicializa con la llamada al método setInterval previa.	
clearTimeout	Cancela el *timeout* que ha sido inicializado con el método setTimeout.	clearTimeout (intervalID)	intervalID se inicializa con la llamada al método setTimeout previa.	
close	Cierra la ventana especificada.	close()		
confirm	Abre una caja de diálogo con un mensaje y los botones *Aceptar* y *Cancelar*.	confirm("mensaje")	String	*True* si el usuario pulsa *Aceptar* y *false* si pulsa *Cancelar*.
focus	Asigna el foco a la ventana o *frame* especificado.	focus()		
moveBy	Mueve la ventana a la posición relativa, respecto a su posición actual, el número de píxeles especificados.	moveBy(horizontal, vertical)	Horizontal: número de píxeles para el desplazamiento horizontal. Vertical: número de píxeles para el desplazamiento vertical.	
moveTo	Mueve la esquina superior izquierda de la ventana a la posición absoluta de la pantalla especificada en píxeles.	moveTo(x, y)	x: coordenada x y: coordenada y	
open	Abre una nueva ventana del navegador.	open(URL, nombre, características)	URL: cadena de texto que especifica la URL que se va a visualizar en la nueva ventana. Nombre: texto que indica el nombre de la ventana.	
print	Hace que aparezca el diálogo de impresión.	print()		

Nombre	Descripción	Sintaxis	Parámetros	Retorno
prompt	Abre una caja de diálogo con un mensaje y un campo de entrada de texto.	prompt(mensaje, [texto])	Mensaje: texto que muestra la caja de diálogo. Texto: opcional, texto que por defecto sale en el campo de entrada de texto.	Dato introducido por el usuario.
resizeBy	Redimensiona la ventana moviendo la esquina inferior derecha según los valores relativos especificados.	resizeBy(horizontal, vertical)	Horizontal: número de píxeles que hay que sumar o restar a la dimensión horizontal. Vertical: número de píxeles que hay que sumar o restar a la dimensión vertical.	
resizeTo	Redimensiona la ventana en los valores especificados.	resizeTo(ancho, alto)	Ancho: ancho en píxeles de la ventana. Alto: altura en píxeles de la ventana.	
scrollBy	Hace un *scroll* del área de la ventana, según los valores relativos a la posición actual.	scrollBy(x,y)	x: número de píxeles que hay que sumar o restar para el *scroll* horizontal. y: número de píxeles que hay que sumar o restar para el *scroll* vertical.	
scrollTo	Hace un *scroll* del área de la ventana, según los valores absolutos indicados.	scrollTo(x, y)	x: coordenada x en píxeles, del área que se va a visualizar. y: coordenada y en píxeles, del área que se va a visualizar.	
setInterval	Ejecuta una función cada vez que transcurre el tiempo especificado y hasta que se ejecuta el método clearInterval.	setInterval(expresión, tiempo) o bien setInterval(función, tiempo, param1, ..., paramN)	Tiempo: número de milisegundos	Identificador
setTimeout	Ejecuta una función una vez transcurrido el tiempo especificado.	setTimeout(expresión, tiempo) o bien setTimeout(función, tiempo, param1, ..., paramN)	Tiempo: número de milisegundos.	Identificador

El uso de algunos de los métodos anteriores necesita de la activación o desactivación de ciertas propiedades relacionadas con la seguridad del navegador; esto significa que es posible que no funcionen debido a restricciones de seguridad.

3.1.1. El método open

Cuando se abre una ventana, se puede especificar la dirección URL, el nombre, el tamaño, los botones y todo un conjunto de atributos que definen el aspecto de la ventana.

La sintaxis del método open es:

```
open(url, nombre, características, reemplazar)
```

donde:

- url es la dirección que indica el documento que se cargará en la ventana;
- nombre es el nombre de la ventana (que se utilizará para referenciarla posteriormente);
- características es una cadena delimitada por comas que indica un conjunto de propiedades de la ventana que se va a crear, y
- reemplazar es un valor lógico y opcional que indica si la URL especificada debe reemplazar el contenido de la ventana actual.

Un primer ejemplo de este método sería:

```
ventanaUoc = open("http://www.uoc.edu", "UOC", "height=300, width=200");
```

De la misma manera que se puede abrir una ventana, es posible cerrarla utilizando el método close():

```
ventanaUoc.close();
```

Pero este método solo puede cerrar ventanas que hayan sido creadas previamente desde el mismo ámbito, y en algunos navegadores, por políticas de seguridad, no se puede cerrar la ventana principal.

La lista de **opciones** del método es muy amplia, a continuación se presentan las más "interesantes":

- **alwaysLowered**: valor booleano que indica que la nueva ventana quedará por debajo del resto de ventanas;
- **alwaysRaised**: valor booleano que indica que la nueva ventana quedará por encima del resto de ventanas;
- **dependent**: valor booleano que indica que la ventana creada es dependiente de la ventana padre. Esto implica que al cerrar una ventana se cierran todas las dependientes de esta;
- **directories**: valor booleano que indica que la nueva ventana contendrá la botones de directorio estándar;
- **height**: valor en píxeles que especifica el alto de la ventana;
- **hotkeys**: valor booleano que indica si las teclas rápidas del navegador están habilitadas en la nueva ventana;

- **innerHeight**: valor en píxeles que especifica el alto del contenido de la ventana;
- **innerWidth**: valor en píxeles que especifica el ancho del contenido de la ventana;
- **left**: valor en píxeles que especifica la coordenada x en la que aparecerá la nueva ventana;
- **location**: valor booleano que indica si la barra de direcciones debe mostrarse en la ventana;
- **menubar**: valor booleano que indica si la nueva ventana contendrá la barra de menú;
- **outerHeight**: valor en píxeles que especifica la dimensión vertical de los bordes de la nueva ventana;
- **resizable**: valor booleano que indica si el usuario podrá cambiar el tamaño de la nueva ventana;
- **scrollbars**: valor booleano que indica si la nueva ventana contendrá las barras de *scroll* cuando el documento exceda los límites de la ventana;
- **status**: valor booleano que indica si la nueva ventana contendrá la barra de estado;
- **titlebar**: valor booleano que indica si la nueva ventana contendrá la barra de título;
- **toolbar**: valor booleano que indica si la nueva ventana contendrá la barra de herramientas estándar;
- **top**: valor en píxeles que especifica la coordenada "y" en la que aparecerá la nueva ventana;
- **width**: valor en píxeles que especifica el ancho de la ventana, y
- **z-lock**: valor booleano que especifica si no se podrá cambiar el orden de pila relativo a otras ventanas incluso si esta obtiene el foco.

El siguiente ejemplo abre una ventana con las características que se usan más comúnmente:

```
function abrir(){
   url= "http://www.uoc.edu";
   caract = "left=50, top=50, status=yes, menubar=yes, toolbar=no,
```

```
    width=590, height=250, directory=no, resize=no, scrollbars=yes";
    return window.open(url,"Ejemplo apertura",caract);
}
```

3.1.2. Cajas de diálogo

Existen **tres métodos** que crean las clásicas cajas de diálogo o mensajes en ventanas modales:

- **alert()**: crea una ventana con un mensaje y el botón *Aceptar* que cierra la ventana;

- **confirm()**: crea una ventana que muestra un mensaje y espera que el usuario responda pulsando el botón *Aceptar* o *Cancelar*. Este devuelve el valor *true* si se ha pulsado el botón *Aceptar*, y *false* si se ha pulsado *Cancelar*;

- **prompt()**:crea una ventana que muestra un mensaje y solicita datos al usuario a partir de un campo de texto. Al igual que confirm(), dispone de los botones *Aceptar* y *Cancelar*; en el caso de presionar *Cancelar*, el método devolverá el valor *null*, en caso contrario, devolverá el contenido del campo de texto introducido por el usuario.

El siguiente ejemplo muestra un caso de uso de los tres métodos:

```
var entrada = prompt("Entra un dato: ", 0);
alert("El dato introducido es: " + entrada);
var respuesta = confirm("¿Desea cerrar la ventana?");
if (respuesta==true)window.close();
```

El código anterior se interpreta perfectamente si se observa el resultado en las figuras siguientes:

Figura 10. Caja de diálogo prompt

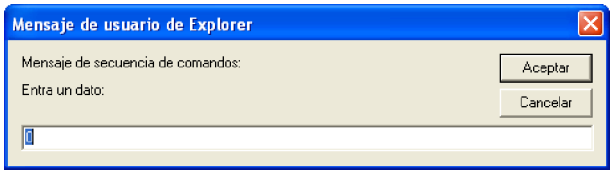

Figura 11. Caja de diálogo alert

Figura 12. Caja de diálogo confirm

3.1.3. Los métodos setTimeout y clearTimeout

En el siguiente ejemplo, se va a crear un reloj, para lo que se van a emplear los métodos setTimeout() y clearTimeout():

```
var timerID = null;
var timerRunning = false;
function stopReloj(){
   if(timerRunning)
      clearTimeout(timerID);
   timerRunning = false;
}
function startReloj(){
   stopReloj();//Hay que asegurar que el reloj está parado
   ver();
}
function ver(){
   var now = new Date();
   var hours = now.getHours();
   var minutes = now.getMinutes();
   var seconds = now.getSeconds();
   var timeValue = "" + ((hours > 12) ? hours - 12 : hours);
   timeValue += ((minutes < 10) ? ":0" : ":") + minutes;
   timeValue += ((seconds < 10) ? ":0" : ":") + seconds;
   timeValue += (hours >= 12) ? " P.M." : " A.M.";
   document.reloj.face.value = timeValue;
   timerID = setTimeout("ver()",1000);
   timerRunning = true;
}
```

El documento HTML solo necesitará implementar lo siguiente:

- Realizar una llamada a la función startReloj().
- Disponer de un formulario y un campo de texto que se llamen “reloj” y “face”, respectivamente.

La base del funcionamiento de reloj está en la llamada recursiva cada segundo de la función ver(); de esta manera, a cada segundo se va actualizando la información de la hora en el campo de texto del formulario.

3.1.4. El método moveBy

En el siguiente ejemplo, se puede observar el uso de los métodos moveBy y setInterval de manera coordinada, lo que provoca el desplazamiento por la pantalla de la ventana recién creada:

```
var v1;
function abrir(){
   var caract = "left=200, top=10, width=400, height=210, statusbar=no, menubar=no, directory=no,
   resize=no, scrollbars=no";
   v1=window.open("Movimiento.html","Ventana móvil",caract);
   ini_mover()
}

function ini_mover(){
   setInterval('mover()',500);
}

function mover(){
   v1.moveBy(5,10);
}
```

La llamada a la función abrir() debe realizarse desde la página HTML, ya que en primer lugar crea una nueva ventana y, a continuación, llama a la función que inicia el movimiento.

3.2. El objeto Location

El objeto Location proporciona acceso a la dirección URL y a porciones de la URL de manera estructurada. Asignar una cadena a un objeto Location provoca que el navegador analice la cadena como una dirección de internet, actualice las propiedades del objeto y establezca la cadena como la propiedad *href* del objeto.

Una URL tiene la siguiente estructura:

```
protocol//host:port/pathname#hash?search
```

Por ejemplo:

```
http://www.uoc.edu:8080/assist/extensions.html#topic1?x=7&y=2
```

El significado de cada una de las partes que componen la URL es el siguiente:

- protocol: representa el inicio de la URL, es decir el protocolo web que incluye los dos puntos (http:);

- host: representa el dominio o la dirección IP del ordenador anfitrión en la red (www.uoc.edu);

- port: representa el puerto de comunicación que el servidor usa para las comunicaciones (8080);

- pathname: representa la ruta del documento (assist/extensions/);

- hash: representa el ancla en la URL, incluido el signo # (solo para http);

- search: representa la información para una búsqueda (*query*). Incluye el signo ?, que indica el inicio (solo para http). Cada elemento de la búsqueda está compuesto por el nombre de la variable y su valor, y cada elemento se separa del siguiente por el signo &.

Cuando se asigna un valor a la propiedad *location* de un objeto, JavaScript crea un objeto Location y asigna este valor a la propiedad *href* del objeto Location. Por tanto, las dos formas que se muestran a continuación son equivalentes:

```
window.location.href="http://www.uoc.edu";
window.location="http://www.uoc.edu";
```

El objeto Location hereda directamente del objeto Window. Si se hace referencia al objeto Location sin especificar una ventana, la información de este estará asociada a la ventana actual.

A continuación, se muestran en una tabla las principales propiedades del objeto Location:

Tabla 10. Propiedades del objeto Location

Nombre	Descripción	Ejemplo
hash	Nombre de ancla en la URL.	alert(window.**location.hash**)
host	Nombre del servidor o del dominio que forma parte del parámetro hostname.	window.**location.host** = "www.uoc.edu"
hostname	Contiene el nombre completo del servidor, el host y el puerto.	window.**location.host** = "www.uoc.edu:8080"
href	Especifica la URL completa.	window.**location.href**="http://www.uoc.es/"
pathname	Especifica la ruta del documento.	window.**location.pathname** = "/js/ejemplos/reloj.html"
port	Puerto de comunicaciones que usa el servidor.	window.location.port = 80
protocol	Inicio de la URL que especifica la forma de acceso.	window.location.protocol = "http:"

Nombre	Descripción	Ejemplo
search	Especifica la búsqueda en la URL.	window.**location.search** = "?cp=2&scp=4"

La siguiente tabla muestra los principales métodos del objeto Location:

Tabla 11. Métodos del objeto Location

Nombre	Descripción	Sintaxis	Parámetros
assign	Es equivalente a modificar la URL mediante location.href	assign(url)	url: cadena de texto.
reload	Actualiza el documento.	reload([forceGet])	Opcional. Si es true fuerza la carga del documento con el método GET.
replace	Carga la URL especificada en la ventana y sustituye a la que había.	replace(url)	url: cadena de texto.

Respecto a los métodos anteriores, es importante tener en cuenta que la URL de una página cargada con el método replace substituye a la anterior en el historial de páginas visitadas, por lo que al reemplazar una página la referencia a esta es substituida en el historial.

En el siguiente ejemplo, se puede observar el uso de propiedades y métodos de este objeto:

```
<html>
<head>
<title>Uso del objeto location</title>
<script type="text/javascript">
   function mostrarImagen(img) {
      document.images[0].src=img;
   }
</script>
</head>
<body>
<form name="miForm">
   <h2>Escoja un regalo:</h2>
   <p>
      <input type="radio" name="imagen" value="imagen1" checked
      onclick="mostrarImagen('img/regalo1.gif')"> Verde <br>
      <input type ="radio" name ="imagen" value ="imagen2"
      onclick="mostrarImagen('img/regalo2.gif')"> Amarillo <br>
      <input type ="radio" name ="imagen" value ="imagen3"
      onclick="mostrarImagen('img/regalo3.gif')"> Rojo
   </p>
   <p>
      <img name="mi_imagen" SRC="img/regalo1.gif" align="center">
      <input type ="button" value="Actualizar" onclick="window.location.reload()">
      <input type="button" value="Ver regalo"
```

```
        onclick="window.location.replace('regalo.htm')" name="button">
    </p>
</form>
</body>
</html>
```

En el ejemplo anterior, se ha insertado un botón en el formulario que realiza una recarga de la página HTML a partir del método reload() y un nuevo botón *Ver regalo*, que substituye la página cargada por una nueva "regalo.htm", que supuestamente mostrará el regalo elegido. También se necesita tener creada la página "regalo.htm" y las imágenes "regalo1.gif", "regalo2.gif" y "regalo3.gif" en una subcarpeta con el nombre "img".

3.3. El objeto History

El navegador almacena un *array* con las direcciones web visitadas en el objeto History, de manera que este proporciona propiedades y métodos que permitirán desde JavaScript acceder a estas direcciones y visitar estas páginas web.

Al objeto History se accede mediante la propiedad *history* de Window (window.history), ya que es un objeto que hereda directamente de Window.

Tal como se ha comentado, mantiene el historial de páginas visitadas, guardando las URL en un *array* de objetos y, por lo tanto, si se accede a una posición del *array*, por ejemplo history[2], se obtendrá la URL visitada en tercer lugar (la primera posición de un *array* en JavaScript es la 0). Se debe tener en cuenta que, por motivos de seguridad, es posible que ciertas propiedades estén restringidas por defecto en algunos navegadores.

Las principales propiedades del objeto son las siguientes:

Tabla 12. Propiedades del objeto History

Nombre	Descripción	Ejemplo
length	Número de elementos del *array*. Es de solo lectura.	alert(history.length)
state	Devuelve el valor del estado del objeto.	alert(history.state)

Los principales métodos del objeto son los siguientes:

Tabla 13. Métodos del objeto History

Nombre	Descripción	Sintaxis	Parámetros
back	Carga la anterior URL del historial.	back()	
forward	Carga la siguiente URL del historial.	forward()	

Nombre	Descripción	Sintaxis	Parámetros
go	Carga una URL del historial.	go(pos) go(url)	pos: entero que representa la posición relativa en el historial. url: string que representa una url del historial.

Respecto a los métodos anteriores, se deben tener en cuenta los siguientes detalles:

- El método back tiene el mismo efecto que usar el método go de la forma: history.go(-1).
- El método forward tiene el mismo efecto que usar el método go de la forma: history.go(1).
- La forma del método go, go(0), tiene como efecto la actualización de la página actual.

A continuación, se presenta un ejemplo en el que se puede observar el uso de los métodos del objeto:

```
<html>
<head>
</head>
<body>
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
      <td bgcolor="#000000">
         <div align="center"><a href="javascript:history.back()">Anterior</a></div>
      </td>
      <td bgcolor="#000000">
         <div align="center"><a href="javascript:history.forward()">Siguiente</a></div>
      </td>
      <td bgcolor="#000000">
         <div align="center"><a href="javascript:history.go(0)">Actualizar</a></div>
      </td>
</tr>
</table>
<p align="center">P&aacute;gina 1</p>
<div align="center">
   <p><a href="http://www.uoc.edu">P&aacute;gina 2</a></p>
   <p><a href="http://www.google.com">P&aacute;gina 3</a></p>
</div>
</body>
</html>
```

3.4. El objeto Screen

El objeto Screen contiene propiedades de solo lectura, que suministran información acerca de la pantalla del usuario. Se trata de un objeto hijo de Window.

A continuación, se presentan las principales propiedades del objeto:

Tabla 14. Propiedades del objeto Screen

Nombre	Descripción
availHeight	Altura en píxeles de la pantalla sin contar las posibles utilidades que pueda mostrar el sistema en pantalla, como las barras de herramientas.
availWidth	Ancho en píxeles de la pantalla sin contar las posibles utilidades que pueda mostrar el sistema en pantalla, como las barras de herramientas.
colorDepth	Si existe una paleta en uso, indica la profundidad de color en bits por píxel; en otro caso, deriva de screen.pixelDepth.
height	Altura de la pantalla en píxeles.
pixelDepth	Resolución de la pantalla en bits por píxel.
width	Ancho de la pantalla en píxeles.

El objeto Screen no dispone de métodos.

3.5. El objeto Navigator

El objeto Navigator contiene la información del navegador que se está usando. Estas propiedades son utilizadas generalmente para la detección del navegador, aunque también proporcionan una serie de detalles sobre la configuración del usuario, como el idioma de preferencia y el sistema operativo.

Este objeto es descendiente directo del objeto Window y todas sus propiedades son de solo lectura. A continuación, se pueden consultar las principales propiedades del objeto en la tabla siguiente:

Tabla 15. Propiedades del objeto Navigator

Nombre	Descripción	Ejemplo
appCodeName	Nombre del código del navegador.	document.write("El código de su navegador es " + navigator.appCodeName)
appName	Nombre del navegador.	document.write("El nombre de su navegador es " + navigator.appName)
appVersion	Versión del navegador.	document.write("La versión de su navegador es " + navigator.appVersion)
cookieEnabled	Indica si el navegador tiene activas las cookies.	if (navigator.cookieEnabled) alert("Las cookies están activadas en el navegador")

Nombre	Descripción	Ejemplo
mimeTypes	*Array* de todos los tipos MIME soportados por el navegador.	
platform	Sistema operativo sobre el que se ejecuta el navegador.	alert(navigator.platform) Podría mostrar datos tales como: Win32, Win16, Mac68k, MacPPC, etc.
plugins	*Array* de *plugins* instalados en el navegador.	
userAgent	Valor de la cabecera user-agent enviada en el protocolo HTTP, del cliente al servidor.	document.write("La cabecera user-agent enviada en el protocolo HTTP es " + navigator.userAgent)
vendor	Cadena que contiene información de la marca comercial del navegador.	

La siguiente tabla muestra el método más usado de los disponibles en el objeto:

Tabla 16. Métodos del objeto Navigator

Nombre	Descripción	Sintaxis	Retorno
javaEnabled	Testea si la opción Java está activada.	javaEnabled()	*True* si está activo java y *false* en caso contrario.

En el siguiente ejemplo, se muestra el uso de las propiedades del objeto:

```
<html>
<head>
</head>
<body>
<h1>Propiedades del navegador</h1>
<script type="text/javascript">
   document.write("Nombre código: <b>" + navigator.appCodeName + "</b><br>")
   document.write("Nombre: <b>" + navigator.appName + "</b><br>")
   document.write("Versión: <b>" + navigator.appVersion + "</b><br>")
   document.write("Idioma: <b>" + navigator.language + "</b><br>")
   document.write("Tipos MIME: <b>" + navigator.mimeTypes + "</b><br>")
   document.write("Plataforma: <b>" + navigator.platform + "</b><br>")
   document.write("Plugins: <b>" + navigator.plugins + "</b><br>")
   document.write("Cabecera URL: <b>" + navigator.userAgent + "</b><br>")
</script>
</body>
</html>
```

3.6. El objeto MimeType

El objeto MimeType (*multipart internet mail extension*) representa el tipo MIME soportado por el cliente. Se puede acceder a este objeto mediante el *array* MimeType del objeto Navigator o desde el objeto Plugin:

```
navigator.mimeTypes[index]
```

De esta manera, el programador puede consultar si un tipo MIME en particular está soportado y, si es así, extraer información sobre el objeto Plugin que lo controla. Las propiedades de este objeto son del tipo solo lectura y se presentan en la tabla siguiente:

Tabla 17. Propiedades del objeto MimeType

Nombre	Descripción	Ejemplo
description	Descripción del tipo MIME.	navigator.mimeTypes["image/jpeg"].description El resultado sería: JPEG Image
enabledPlugin	Referencia al objeto plugin configurado por el tipo MIME. Si no hay ninguno, la propiedad vale *null*.	navigator.mimeTypes["image/jpeg"].enabledPlugins
suffixes	String que contiene la lista de extensiones aceptadas por el tipo MIME.	navigator.mimeTypes["image/jpeg"].suffixes El resultado sería: jpeg, jpg, jpe, jfif, pjpeg, pjp
type	Nombre del tipo MIME.	navigator.mimeTypes["image/jpeg"].type El resultado sería: image/jpeg

El siguiente ejemplo muestra las propiedades para cada objeto MimeType del cliente.

```
<html>
<head>
</head>
<body>
<h1>Propiedades de cada objeto mimeType del cliente:</h1>
<script type="text/javascript">
   document.writeln("<table border=1><tr valing=top>"+ "<th align=left>i"+ " <th align=left>type"+
   "<th align=left>description"+ " <th align=left>suffixes"+
   " <th align=left>enabledPlugin.name </tr>")
   for (i=0; i < navigator.mimeTypes.length; i++) {
      document.writeln("<tr valing=top><td tipus='fuoc'>"+i+" <td tipus='fuoc'>"+
      navigator.mimeTypes[i].type+ " <td tipus='fuoc'>"+
      navigator.mimeTypes[i].description+" <td tipus='fuoc'>"+navigator.mimeTypes[i].suffixes)
      if (navigator.mimeTypes[i].enabledPlugin==null) {
         document.writeln("<td tipus='fuoc'>None"+ " </tr>")
      } else {
```

```
            document.writeln("<td tipus='fuoc'>"+navigator.mimeTypes[i].enabledPlugin.name+" </tr>")
        }
    }
    document.writeln("</table>")
</script>
</body>
</html>
```

3.7. El objeto Plugin

Cada objeto Plugin corresponde a un componente instalado en el navegador. Estos objetos están disponibles mediante la propiedad *enabledPlugin* de objetos MimeType. Cada *plugin* proporciona información sobre el componente como su descripción, los tipos MIME que soporta, etc.

Este objeto se suele utilizar para determinar si el navegador soporta un componente *plugin* específico y su versión.

A continuación, se presentan las principales propiedades del objeto:

Tabla 18. Propiedades del objeto Plugin

Nombre	Descripción
description	Descripción del *plugin*. Es de solo lectura.
filename	Nombre del plugin en el disco. Es de solo lectura.
length	Número de elementos del *array* de objetos MimeType. Es de solo lectura.
name	Nombre del *plugin*. Es de solo lectura.

El siguiente ejemplo muestra las propiedades para cada *plugin* instalado en el cliente.

```
<html>
<head>
</head>
<body>
<b>Propiedades para cada conector instalado en el cliente:</b><p>
<script type="text/javascript">
   document.writeln("<table botder=1><tr valign=top>"+ "<th valign=left>i"
   + "<th valign=left>name" + "<th valign=left>filename" + "<th valign=left>description"+
   "<th valign=left># of types</tr>")
   for (i=0; i < navigator.plugins.length; i++) {
      document.writeln("<th valign=top><td tipus='fuoc'>"+i+ "<td tipus='fuoc'>",
      navigator.plugins[i].name+ "<td tipus='fuoc'>"+ navigator.plugins[i].filename+
```

```
        "<td tipus='fuoc'>"+navigator.plugins[i].description+ "<td tipus='fuoc'>"+
        navigator.plugins[i].length+ "</tr>")
    }
    document.writeln("</table>")
</script>
</body>
</html>
```