Árbol de nodos BOM

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.

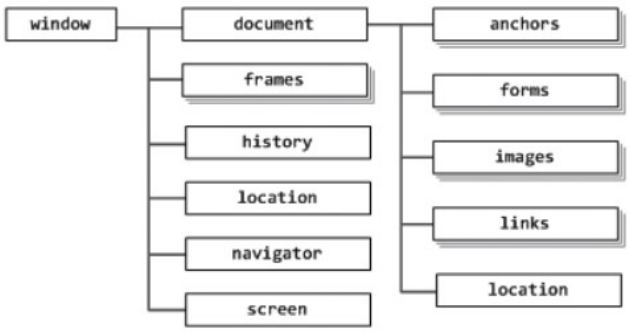
Mediante BOM, es posible redimensionar y mover la ventana del navegador, modificar el texto que se muestra en la barra de estado y realizar muchas otras manipulaciones no relacionadas con el contenido de la página HTML.

El mayor inconveniente de BOM es que, al contrario de lo que sucede con DOM, ninguna entidad se encarga de estandarizarlo o definir unos mínimos de interoperabilidad entre navegadores.

Algunos de los elementos que forman el BOM son los siguientes:

- Crear, mover, redimensionar y cerrar ventanas de navegador.
- Obtener información sobre el propio navegador.
- Propiedades de la página actual y de la pantalla del usuario.
- Gestión de cookies.
- Objetos ActiveX en Internet Explorer.

El BOM está compuesto por varios objetos relacionados entre sí. El siguiente esquema muestra los objetos de BOM y su relación:



En el esquema anterior, los objetos mostrados con varios recuadros superpuestos son arrays. El resto de objetos, representados por un rectángulo individual, son objetos simples. En cualquier caso, todos los objetos derivan del objeto *window*.

El objeto window

El objeto *window* representa la ventana completa del navegador. Mediante este objeto, es posible mover, redimensionar y manipular la ventana actual del navegador. Incluso es posible abrir y cerrar nuevas ventanas de navegador.

Como todos los demás objetos heredan directa o indirectamente del objeto *window*, no es necesario indicarlo de forma explícita en el código JavaScript. En otras palabras:

```
window.forms[0] === forms[0]
window.document === document
```

BOM define cuatro métodos para manipular el tamaño y la posición de la ventana:

- *moveBy(x, y)* desplaza la posición de la ventana x píxel hacia la derecha y y píxel hacia abajo. Se permiten desplazamientos negativos para mover la ventana hacia la izquierda o hacia arriba.
- *moveTo(x, y)* desplaza la ventana del navegador hasta que la esquina superior izquierda se encuentre en la posición (x, y) de la pantalla del usuario. Se permiten desplazamientos negativos, aunque ello suponga que parte de la ventana no se visualiza en la pantalla.
- *resizeBy(x, y)* redimensiona la ventana del navegador de forma que su nueva anchura sea igual a (anchura_anterior + x) y su nueva altura sea igual a (altura_anterior + y). Se pueden emplear valores negativos para reducir la anchura y/o altura de la ventana.
- *resizeTo(x, y)* redimensiona la ventana del navegador hasta que su anchura sea igual a x y su altura sea igual a y. No se permiten valores negativos.

Los navegadores son cada vez menos permisivos con la modificación mediante JavaScript de las propiedades de sus ventanas. De hecho, la mayoría de navegadores permite a los usuarios bloquear el uso de JavaScript para realizar cambios de este tipo. De esta forma, una aplicación nunca debe suponer que este tipo de funciones están disponibles y funcionan de forma correcta.

Además de desplazar y redimensionar la ventana del navegador, es posible averiguar la posición y tamaño actual de la ventana. Sin embargo, la ausencia de un estándar para BOM provoca que cada navegador implemente su propio método:

- Internet Explorer proporciona las propiedades window.*screenLeft y window.screenTop* para obtener las coordenadas de la posición de la ventana. No es posible obtener el tamaño de la ventana completa, sino solamente del área visible de la página (es decir, sin barra de estado ni menús). Las propiedades que proporcionan estas dimensiones son *document.body.offsetWidth y document.body.offsetHeight*.

- Los navegadores de la familia Mozilla, Safari y Opera proporcionan las propiedades *window.screenX* y *window.screenY* para obtener la posición de la ventana. El tamaño de la zona visible de la ventana se obtiene mediante *window.innerWidth* y *window.innerHeight*, mientras que el tamaño total de la ventana se obtiene mediante *window.outerWidth* y *window.outerHeight*.

El objeto document

El objeto *document* es el único que pertenece tanto al DOM (como se vio en el capítulo anterior) como al BOM. Desde el punto de vista del BOM, el objeto *document* proporciona información sobre la propia página HTML.

Algunas de las propiedades más importantes definidas por el objeto *document* son:

Propiedad	Descripción
`lastModified`	La fecha de la última modificación de la página
`referrer`	La URL desde la que se accedió a la página (es decir, la página anterior en el array `history`)
`title`	El texto de la etiqueta `<title>`
`URL`	La URL de la página actual del navegador

Las propiedades *title* y *URL* son de lectura y escritura, por lo que además de obtener su valor, se puede establecer de forma directa:

```
// modificar el título de la página
document.title = "Nuevo titulo";

// llevar al usuario a otra página diferente
document.URL = "http://nueva_pagina";
```

Además de propiedades, el objeto *document* contiene varios *arrays* con información sobre algunos elementos de la página:

Array	Descripción
`anchors`	Contiene todas las "anclas" de la página (los enlaces de tipo `<a name="nombre_ancla"></a>`)
`applets`	Contiene todos los applets de la página
`embeds`	Contiene todos los objetos embebidos en la página mediante la etiqueta `<embed>`
`forms`	Contiene todos los formularios de la página
`images`	Contiene todas las imágenes de la página
`links`	Contiene todos los enlaces de la página (los elementos de tipo `<a href="enlace.html"></a>`)

Los elementos de cada array del objeto *document* se pueden acceder mediante su índice numérico o mediante el nombre del elemento en la página HTML. Si se considera por ejemplo la siguiente página HTML:

```
<html>
  <head><title>Pagina de ejemplo</title></head>
  <body>
    <p>Primer parrafo de la pagina</p>
    <a href="otra_pagina.html">Un enlace</a>
    <img src="logo.jpg" name="logotipo"/>
    <form method="post" name="consultas">
      <input type="text" name="id" />
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Para acceder a los elementos de la página se pueden emplear las funciones DOM o los objetos de BOM:

- Párrafo: *document.getElementsByTagName("p")*
- Enlace: *document.links[0]*
- Imagen: *document.images[0] o document.images["logotipo"]*
- Formulario: *document.forms[0] o document.forms["consultas"]*

Una vez obtenida la referencia al elemento, se puede acceder al valor de sus atributos HTML utilizando las propiedades de DOM. De esta forma, el método del formulario se obtiene mediante *document.forms["consultas"].method* y la ruta de la imagen es d*ocument.images[0].src*.

El objeto location

El objeto *location* es uno de los objetos más útiles del BOM. Debido a la falta de estandarización, *location* es una propiedad tanto del objeto *window* como del objeto *document*.

El objeto *location* representa la URL de la página HTML que se muestra en la ventana del navegador y proporciona varias propiedades útiles para el manejo de la URL:

Propiedad	Descripción
`hash`	El contenido de la URL que se encuentra después del signo `#` (para los enlaces de las anclas) `http://www.ejemplo.com/ruta1/ruta2/pagina.html#seccion hash` – `#seccion`
`host`	El nombre del servidor `http://www.ejemplo.com/ruta1/ruta2/pagina.html#seccion host` – `www.ejemplo.com`
`hostname`	La mayoría de las veces coincide con `host`, aunque en ocasiones, se eliminan las `www` del principio `http://www.ejemplo.com/ruta1/ruta2/pagina.html#seccion hostname` – `www.ejemplo.com`
`href`	La URL completa de la página actual `http://www.ejemplo.com/ruta1/ruta2/pagina.html#seccion URL` – `http://www.ejemplo.com/ruta1/ruta2/pagina.html#seccion`
`pathname`	Todo el contenido que se encuentra después del host `http://www.ejemplo.com/ruta1/ruta2 /pagina.html#seccion pathname` – `/ruta1/ruta2/pagina.html`
`port`	Si se especifica en la URL, el puerto accedido `http://www.ejemplo.com:8080/ruta1/ruta2 /pagina.html#seccion port` – `8080` La mayoría de URL no proporcionan un puerto, por lo que su contenido es vacío `http://www.ejemplo.com/ruta1/ruta2/pagina.html#seccion port` – (vacío)
`protocol`	El protocolo empleado por la URL, es decir, todo lo que se encuentra antes de las dos barras inclinadas `// http://www.ejemplo.com/ruta1/ruta2/pagina.html#seccion protocol` – `http:`
`search`	Todo el contenido que se encuentra tras el símbolo `?`, es decir, la consulta o *"query string"* `http://www.ejemplo.com/pagina.php?variable1=valor1&variable2=valor2 search` – `?variable1=valor1&variable2=valor2`

De todas las propiedades, la más utilizada es *location.href*, que permite obtener o establecer la dirección de la página que se muestra en la ventana del navegador.

El objeto navigator

El objeto *navigator* es uno de los primeros objetos que incluyó el BOM y permite obtener información sobre el propio navegador. En Internet Explorer, el objeto *navigator* también se puede acceder a través del objeto *clientInformation*.

Aunque es uno de los objetos menos estandarizados, algunas de sus propiedades son comunes en casi todos los navegadores. A continuación se muestran algunas de esas propiedades:

Propiedad	Descripción
`appCodeName`	Cadena que representa el nombre del navegador (normalmente es `Mozilla`)
`appName`	Cadena que representa el nombre oficial del navegador
`appMinorVersion`	(Sólo Internet Explorer) Cadena que representa información extra sobre la versión del navegador
`appVersion`	Cadena que representa la versión del navegador
`browserLanguage`	Cadena que representa el idioma del navegador
`cookieEnabled`	Boolean que indica si las cookies están habilitadas
`cpuClass`	(Sólo Internet Explorer) Cadena que representa el tipo de CPU del usuario ("x86", "68K", "PPC", "Alpha", "Other")
`javaEnabled`	Boolean que indica si Java está habilitado
`language`	Cadena que representa el idioma del navegador
`mimeTypes`	Array de los tipos MIME registrados por el navegador
`onLine`	(Sólo Internet Explorer) Boolean que indica si el navegador está conectado a Internet
`oscpu`	(Sólo Firefox) Cadena que representa el sistema operativo o la CPU
`platform`	Cadena que representa la plataforma sobre la que se ejecuta el navegador
`plugins`	Array con la lista de plugins instalados en el navegador
`preference()`	(Sólo Firefox) Método empleado para establecer preferencias en el navegador
`product`	Cadena que representa el nombre del producto (normalmente, es `Gecko`)
`productSub`	Cadena que representa información adicional sobre el producto (normalmente, la versión del motor Gecko)
`securityPolicy`	Sólo Firefox
`systemLanguage`	(Sólo Internet Explorer) Cadena que representa el idioma del sistema operativo
`userAgent`	Cadena que representa la cadena que el navegador emplea para identificarse en los servidores
`userLanguage`	(Sólo Explorer) Cadena que representa el idioma del sistema operativo
`userProfile`	(Sólo Explorer) Objeto que permite acceder al perfil del usuario

El objeto *navigator* se emplea habitualmente para detectar el tipo y/o versión del navegador en las aplicaciones cuyo código difiere para cada navegador. Además, se emplea para detectar si el navegador tiene habilitadas las cookies y Java y también para comprobar los plugins disponibles en el navegador.

El objeto screen

El objeto *screen* se utiliza para obtener información sobre la pantalla del usuario. Uno de los datos más importantes que proporciona el objeto *screen* es la resolución del monitor en el que se están visualizando las páginas. Las siguientes propiedades están disponibles en el objeto *screen*:

Propiedad	Descripción
`availHeight`	Altura de pantalla disponible para las ventanas
`availWidth`	Anchura de pantalla disponible para las ventanas
`colorDepth`	Profundidad de color de la pantalla (32 bits normalmente)
`height`	Altura total de la pantalla en píxel
`width`	Anchura total de la pantalla en píxel

La altura/anchura de pantalla disponible para las ventanas es menor que la altura/anchura total de la pantalla, ya que se tiene en cuenta el tamaño de los elementos del sistema operativo como por ejemplo la barra de tareas y los bordes de las ventanas del navegador.

Además de la elaboración de estadísticas de los equipos de los usuarios, las propiedades del objeto *screen* se utilizan por ejemplo para determinar cómo y cuanto se puede redimensionar una ventana y para colocar una ventana centrada en la pantalla del usuario.

El siguiente ejemplo redimensiona una nueva ventana al tamaño máximo posible según la pantalla del usuario:

```
window.moveTo(0, 0);
window.resizeTo(screen.availWidth, screen.availHeight);
```

El objeto history

Este objeto se encarga de almacenar una lista con los sitios por los que se ha estado navegando, es decir, guarda las referencias de los lugares visitados. Se utiliza, sobre todo, para movernos hacia delante o hacia atrás en dicha lista. Estas son sus propiedades:

- **current**. Cadena que contiene la URL completa de la entrada actual en el historial.
- **next**. Cadena que contiene la URL completa de la siguiente entrada en el historial.
- **length**. Entero que contiene el número de entradas del historial (i.e., cuántas direcciones han sido visitadas).
- **previous**. Cadena que contiene la URL completa de la anterior entrada en el historial.

Y estos son algunos de los metodos de este objeto:

- **back().** Vuelve a cargar la URL del documento anterior dentro del historial.
- **forward().** Vuelve a cargar la URL del documento siguiente dentro del historial.
- **go(posicion).** Vuelve a cargar la URL del documento especificado por posicion dentro del historial. posicion puede ser un entero, en cuyo caso indica la posición relativa del documento dentro del historial; o puede ser una cadena de caracteres, en cuyo caso representa toda o parte de una URL que esté en el historial.

Control de tiempos

Al contrario que otros lenguajes de programación, JavaScript no incorpora un método *wait()* que detenga la ejecución del programa durante un tiempo determinado. Sin embargo, JavaScript proporciona los métodos *setTimeout()* y *setInterval()* que se pueden emplear para realizar tareas similares.

El método *setTimeout()* permite ejecutar una función al transcurrir un determinado periodo de *tiempo.* El método *setTimeout()* requiere dos argumentos. El primero es el código que se va a ejecutar o una referencia a la función que se debe ejecutar. El segundo argumento es el tiempo, en milisegundos, que se espera hasta que comienza la ejecución del código. El ejemplo anterior se puede rehacer utilizando una función

```
function muestraMensaje() {
  console.log("Han transcurrido 3 segundos desde que me programaron");
}

setTimeout(muestraMensaje, 3000);
```

Cuando se establece una cuenta atrás, la función *setTimeout()* devuelve el identificador de esa nueva cuenta atrás. Empleando ese identificador y la función *clearTimeout()* es posible impedir que se ejecute el código pendiente:

```
function muestraMensaje() {
  console.log("Han transcurrido 3 segundos desde que me programaron");
}
var id = setTimeout(muestraMensaje, 3000);

// Antes de que transcurran 3 segundos, se decide eliminar la ejecución pen
diente
clearTimeout(id);
```

Además de programar la ejecución futura de una función, JavaScript también permite establecer la ejecución periódica y repetitiva de una función. El método necesario es *setInterval()* y su funcionamiento es idéntico al mostrado para *setTimeout()*.
De forma análoga a *clearTimeout()*, también existe un método que permite eliminar una repetición periódica y que en este caso se denomina *clearInterval()*.

```
function muestraMensaje() {
  console.log("Este mensaje se muestra cada segundo");
}

var id = setInterval(muestraMensaje, 1000);

// Despues de ejecutarse un determinado número de veces, se elimina el inte
rvalo
clearInterval(id);
```