Browser Object Model (BOM)

Introducción

Si en el tema anterior vimos cómo interactuar con la página (DOM) en este veremos cómo acceder a objetos que nos permitan interactuar con el navegador

(Browser Object Model, BOM).

Usando los objetos BOM podemos:

- Abrir, cambiar y cerrar ventanas
- Ejecutar código en cierto tiempo (*timers*)
- Obtener información del navegador
- Ver y modificar propiedades de la pantalla
- Gestionar cookies, …

Timers

Permiten ejecutar código en el futuro (cuando transcurran los milisegundos indicados). Hay 2 tipos:

- **`setTimeout(función, milisegundos)`:** ejecuta la función indicada una sóla vez, cuando transcurran los milisegundos

- **`setInterval(función, milisegundos):`** ejecuta la función indicada cada vez que transcurran los milisegundos, hasta que sea cancelado el *timer*.

  A ambas se le pueden pasar más parámetros tras los milisegundos y serán los parámetros que recibirá la función a ejecutar.

Ambas funciones devuelven un identificador que nos permitirá cancelar la ejecución del código, con:

- **`clearTimemout(identificador)`**
- **`clearInterval(identificador)`**

Ejemplo:

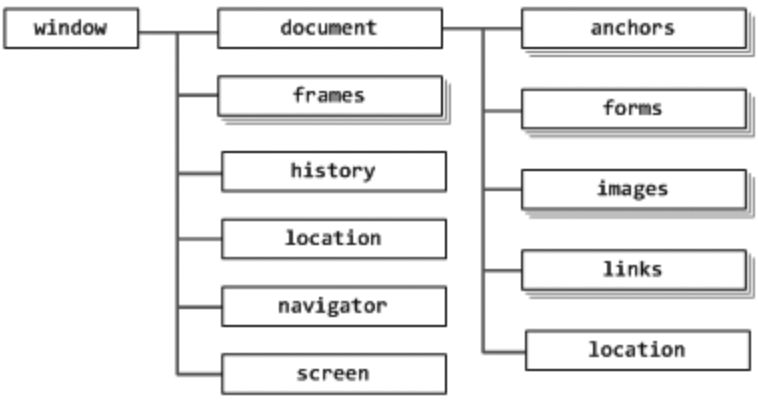
```
let idTimeout=setTimeout(function() {
        alert('Timeout que se ejecuta al cabo de 1 seg.')
}, 1000);

let i=1;
let idInterval=setInterval(function() {
        alert('Interval cada 3 seg. Ejecución nº: '+ i++);
   if (i==5) {
      clearInterval(idInterval);
      alert('Fin de la ejecución del Interval');
        }
}, 3000);
```

EJERCICIO: Ejecuta en la consola cada una de esas funciones

Objetos del BOM

Al contrario que para el DOM, no existe un estándar de BOM pero es bastante parecido en los diferentes navegadores.



Objeto window

Representa la ventana del navegador y es el objeto principal. De hecho puede omitirse al llamar a sus propiedades y métodos, por ejemplo, el método **`setTimeout()`** es en realidad **`window.setTimeout()`.**
Sus principales propiedades y métodos son:

- **`.name`:** nombre de la ventana actual
- **`.status`:** valor de la barra de estado
- **`.screenX/.screenY`:** distancia de la ventana a la esquina izquierda/superior de la pantalla
- **`.outerWidth/.outerHeight`:** ancho/alto total de la ventana, incluyendo la toolbar y la scrollbar
- **`.innerWidth/.innerHeight`:** ancho/alto útil del documento, sin la toolbar y la scrollbar
- **`.open([url], [nombre], [opciones])`:** abre una nueva ventana. Devuelve el nuevo objeto ventana. Las principales opciones son:
  - `.toolbar`: si tendrá barra de herramientas
  - `.location`: si tendrá barra de dirección
  - `.directories`: si tendrá botones Adelante/Atrás
  - `.status`: si tendrá barra de estado
  - `.menubar`: si tendrá barra de menú
  - `.scrollbar`: si tendrá barras de desplazamiento
  - `.resizable`: si se puede cambiar su tamaño
  - `.width=px/.height=px`: ancho/alto
  - `.left=px/.top=px`: posición izq/sup de la ventana
- **`.opener`:** referencia a la ventana desde la que se abrió esta ventana (para ventanas abiertas con *open*)
- **`.close()`:** la cierra (pide confirmación, a menos que la hayamos abierto con open)
- **`.moveTo(x,y)`:** la mueve a las coord indicadas
- **`.moveBy(x,y)`:** la desplaza los px indicados
- **`.resizeTo(x,y)`:** la da el ancho y alto indicados
- **`.resizeBy(x,y)`:** le añade ese ancho/alto
- **`.pageXoffset / pageYoffset`:** scroll actual de la ventana horizontal / vertical
- Otros metodos:

`.back(), .forward(), .home(), .stop(), .focus(), .blur(), .find(), .print(), ...`
NOTA: por seguridad no se puede mover una ventana fuera de la pantalla ni darle un tamaño menor de 100x100 px ni tampoco se puede mover una ventana no abierta con .open() o si tiene varias pestañas.

NOTA: 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.

EJEMPLOS:

```
// Mover la ventana 20 píxel hacia la derecha y 30 píxel hacia
abajo
window.moveBy(20, 30);

// Redimensionar la ventana hasta un tamaño de 250 x 250
window.resizeTo(250, 250);

// Agrandar la altura de la ventana en 50 píxel
window.resizeBy(0, 50);

// Colocar la ventana en la esquina izquierda superior de la ve
ntana
window.moveTo(0, 0);
```

EJERCICIO:

- abre una nueva ventana de dimensiones 500x200px en la posición (100,200)
- escribe en ella (con document.write) un título h1 que diga 'Hola'
- a los 2 segundos se moverá 100 px hacia abajo y 100 a la izquierda
- y se cerrara 3 segundos después.

```
<body>

  <h1>Objeto Window</h1>

<p>Abre "miWindow" y mueve la nueva ventana 100px relativos a su posicion
actual:</p>

<button onclick="abreWin()">Open "miWindow"</button>
```

```
<script>
let miWindow;
var configuracion_ventana = "left=100,top=200,width=500,height=200";

function abreWin() {
   miWindow = window.open("", "", configuracion_ventana);
   miWindow.document.write("<h1>HOLA</h1>");
   let idTimeout=setTimeout(function() {
            miWindow.moveBy(100, 100);
    }, 2000);
   idTimeout=setTimeout(function() {
            miWindow.close();
   }, 3000);
}
</script>
</body>
```

EJERCICIO: Haz que a los 2 segundos de abrir la página se abra un *popup* con un mensaje de bienvenida. Esta ventana tendrá en su interior un botón Cerrar que permitirá que el usuario la cierre haciendo clic en él. Tendrá el tamaño justo para visualizar el mensaje y no tendrá barras de scroll, ni de herramientas, ni de dirección… únicamente el mensaje.

Diálogos

Hay 3 métodos del objeto *window* que nos permiten abrir ventanas de diálogo con el usuario:

- **`window.alert(mensaje)`:** muestra un diálogo con el mensaje indicado y un botón de 'Aceptar'
- **`window.confirm(mensaje)`:** muestra un diálogo con el mensaje indicado y botones de 'Aceptar' y 'Cancelar'. Devuelve *true* si se ha pulsado el botón de aceptar del diálogo y *false* si no.
- **`window.prompt(mensaje [, valor predeterminado])`:** muestra un diálogo con el mensaje indicado, un cuadro de texto (vacío o con el valor predeterminado indicado) y botones de 'Aceptar' y 'Cancelar'. Si se pulsa 'Aceptar' devolverá un *string* con el valor que haya en el cuadro de texto y si se pulsa 'Cancelar' o se cierra devolverá *null*.

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:

- **`.href`:** devuelve la URL actual completa
- **`.protocol`, `.host`, `.port`:** devuelve el protocolo, host y puerto respectivamente de la URL actual
- **`.pathname`:** devuelve la ruta al recurso actual
- **`.reload()`:** recarga la página actual
- **`.assign(url)`:** carga la página pasada como parámetro
- **`.replace(url)`:** ídem pero sin guardar la actual en el historial

EJERCICIO: Ejecuta en la consola

- carga la página de Google usando el objeto
  >location.assign("https://www.google.es")
- probar los comandos anteriores.

Objeto history

Permite acceder al historial de páginas visitadas y navegar por él:

- **`.length`:** muestra el número de páginas almacenadas en el historial
- **`.back()`:** vuelve a la página anterior
- **`.forward()`:** va a la siguiente página
- **`.go(num)`:** se mueve *num* páginas hacia adelante o hacia atrás (si *num* es negativo) en el historial

EJERCICIO: desde la consola probar los anteriores comandos.

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.

- **.appName:** Cadena que representa el nombre oficial del navegador.
- **.appVersion:** Cadena que representa la versión del navegador.
- **.language:** Cadena que representa el idioma del navegador.
- ….

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.

- **.availHeight:** Altura de pantalla disponible para las ventanas.
- **.availWidth:** Anchura de pantalla disponible para las ventanas.
- **.colorDepht:** Profundidad de color de la pantalla.
- **.height:** Altura total de la pantalla en píxel.
- **.width:** Anchura total de la pantalla en píxel.

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);
```

EJERCICIO: obtén desde la consola todas las propiedades width/height y availWidth/availHeight del objeto *screen*. Compáralas con las propiedades innerWidth/innerHeight y outerWidth/outerHeight de *window*.