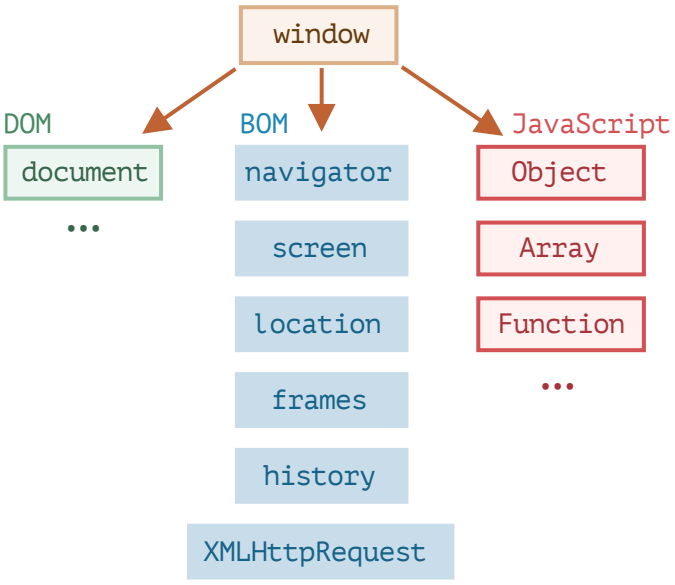
JavaScript 05 – Browser Object Model (BOM) 1
Introducción 1
Timers 1
Objetos del BOM 3
Objeto window 3
Diálogos 4
Objeto location 4
Objeto history 5
Otros objetos 5

JavaScript 05 – Browser Object Model (BOM)

Introducción

Hemos visto cómo interactuar con la página (DOM), ahora veremos cómo acceder a objetos que nos permitan interactuar con el navegador utilizando el BOM (*Browser Object Model*).



Usando los objetos BOM podemos:

- Abrir, cambiar y cerrar ventanas.
- Ejecutar código transcurrido 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 especificada **una sola vez**, cuando transcurran los milisegundos indicados.
- `setInterval(función, milisegundos)`: ejecuta la función especificada **cada vez que transcurran** los milisegundos indicados, hasta que sea cancelado el *timer*.

Se pueden pasar más parámetros a las funciones, se ponen 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:

- `clearTiemout(identificador)`
- `clearInterval(identificador)`

Ejemplo:

```
const idTimeout = setTimeout(() => console.log('Timeout que se ejecuta al cabo de 1
seg.'), 1000);

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

EJERCICIO: Prueba a ejecutar cada una de esas funciones.

En lugar de definir la función a ejecutar podemos llamar a una función que ya exista:

```
function showMessage() {
    console.log('Timeout que se ejecuta al cabo de 1 seg.')
}

const idTimeout=setTimeout(showMessage, 1000);
```

Pero en ese caso hay que poner sólo el nombre de la función, sin `()`, ya que si los ponemos se ejecutaría la función en ese momento y no transcurrido el tiempo indicado.

Si necesitamos pasar algún parámetro a la función, los añadiremos como parámetros de `setTimeout` o `setInterval` después del intervalo.

Ejemplo:

```
function showMessage(msg) {
    alert(msg)
  }

const idTimeout = setTimeout(showMessage, 1000, 'Timeout que se ejecuta al cabo de 1
seg.');
```

Otro ejemplo:

```
function myCallback(a, b) {
    // Tu código debe ir aquí
    // Los parámetros son totalmente opcionales
    console.log(a);
    console.log(b);
}

const intervalID = setInterval(myCallback, 500, 'parámetro 1', 'parámetro 2');
```

Objetos del BOM

Al contrario que para el DOM, no existe un estándar de BOM pero sus objetos (window, location, history, etc) son bastante parecidos 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
- `.statusbar`: 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 de .open()** (lista de ítems separados por comas sin espacios) 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

  Más información en https://www.w3schools.com/jsref/met_win_open.asp

- `.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)`: especifica el ancho y alto indicados
- `.resizeBy(x,y)`: añade esos pixeles de ancho/alto
- `.pageXoffset / pageYoffset`: scroll actual de la ventana horizontal / vertical
- Otros métodos: `.back()`, `.forward()`, `.home()`, `.stop()`, `.focus()`, `.blur()`, `.find()`, `.print()`, …

Más información en https://www.w3schools.com/jsref/obj_window.asp

NOTA: por seguridad no se puede mover una ventana fuera de la pantalla, ni darle un tamaño menor de 100x100 px. Tampoco se puede mover una ventana no abierta con .open(), o si tiene varias pestañas.

EJEMPLO:

a) Abrir una nueva ventana de dimensiones 500x200px en la posición (200,100)
b) Escribir en la nueva ventana (con document.write) un título H1 que diga 'Hola javascritos del Claudio'.
c) Al hacer clic sobre un botón de la ventana inicial, que la ventana se desplace 40 px a la izquierda y 50 hacia abajo
d) Al hacer clic sobre otro botón de la ventana incicial, que se cierre la nueva ventana.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="">
        <input type="button" value="Deslazar" onclick="desplazar()">
        <input type="button" value="Cerrar" onclick="cerrar()">
    </form>
    <script src="js/bom1.js"></script>
</body>
</html>
```

```javascript
function desplazar(){
    nuevaVentana.moveBy(40,50);
}

const cerrar = ()=> {
    nuevaVentana.close();
}

const nuevaVentana=window.open("", "_blank", "width= 500,height=200,left=200,top=100");
nuevaVentana.document.write("<h1>Hola javascritos del Claudio</h1>");
```

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 ya conocemos y 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

Contiene información sobre la URL actual del navegador y podemos modificarla. Sus principales propiedades y métodos son:

- `.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:

a) muestra la ruta completa de la página actual

b) muestra el servidor de esta página

c) carga la página de Google usando el objeto *location*

```
<script>
    console.log( window.location.href);
    console.log(location.host);
    console.log(location.assign('https://www.google.es/'));
</script>
```

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 en el historial (si *num* es positivo), o hacia atrás (si *num* es negativo).

EJERCICIO: Vuelve a la página anterior

Otros objetos

Los otros objetos que incluye BOM son:

- document: el objeto *document* que hemos visto en el DOM
- navigator: nos informa sobre el navegador y el sistema en que se ejecuta
  - `.userAgent`: muestra información sobre el navegador que usamos
  - `.plataform`: muestra información sobre la plataforma sobre la que se ejecuta
  - …
- screen: nos da información sobre la pantalla
  - `.width`/`.height`: ancho/alto total de la pantalla (resolución)
  - `.availWidth`/`.availHeight`: igual pero excluyendo la barra del S.O.
  - …

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