

ÍNDICE DE CONTENIDO

1. Document Object Model (DOM)

1.1. Introducción

1.2. Modificando atributos

1.3. Funciones de JavaScript para localizar elementos en el DOM

1.3.1. getElementById(identificador)

1.3.2. getElementsByTagName(etiqueta)

1.3.3. getElementsByName(nombre)

1.4. Funciones para crear/eliminar nodos

1.4.1. removeChild(nodo)

1.4.2. appendChild(nodo)

2. Browser Object Model (BOM)

2.1. Introducción a BOM

2.2. Objeto window

2.3. Control de tiempos

2.4. Objeto document

2.5. Objeto location

2.6. Objeto navigator

2.7. Objeto screen

Desarrollo Web en Entorno Cliente

1.- DOCUMENT OBJECT MODEL (DOM)

1.1.- Introducción

El llamado DOM (Document Object Model) es un modelo que permite tratar un documento Web XHTML como si fuera XML, navegando por los nodos existentes que forman la página, pudiendo manipular sus atributos e incluso crear nuevos elementos.

Para más información general de DOM:

- https://es.wikipedia.org/wiki/Document_Object_Model
- http://www.w3schools.com/js/js_htmlDOM.asp

Usando JavaScript para navegar en el DOM podemos acceder a todos los elementos XHTML de una página. Esto nos permite cambiar dinámicamente el aspecto de nuestras páginas Web.

En este tema vamos a estudiar las principales funciones de JavaScript para modificar el DOM.

1.2.- Modificando atributos

Cuando obtengamos algún elemento con las funciones que estudiaremos más adelante, podemos manipular los atributos de dicho elemento de la siguiente forma.

```
let elemento=document.getElementById("miElemento");
elemento.innerHTML="El html interno a cambiar de ese elemento";
```

Los atributos disponibles a modificar pueden depender de cada elemento.

1.3.- Funciones de javascript para localizar elementos en el dom

Las funciones aquí estudiadas normalmente se usan sobre el elemento “document”, ya que así se aplican a todo el documento.

Aun así, pueden usarse en cualquier nodo XHTML, entonces la búsqueda se realizaría no en todo en el documento, sino en el sub-árbol formado por el elemento en sí y sus hijos.

Desarrollo Web en Entorno Cliente

1.3.1.- *getElementById (identificador)*

Esta función devuelve un elemento DOM del subárbol cuyo identificador sea el indicado en la cadena "identificador".

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

Ejemplo:

```
let myDiv = document.getElementById("miDiv");
console.log("El html de miDiv es "+myDiv.innerHTML);
```

1.3.2.- *getElementsByTagName (etiqueta)*

Esta función devuelve un array con todos los elementos DOM del subárbol cuya etiqueta XHTML sea la indicada en la cadena "etiqueta".

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

Ejemplo:

```
let myDiv = document.getElementById("miDiv")
let losP = myDiv.getElementsByTagName("p");
let num = losP.length;
console.log("Hay " + num + " <p> elementos en el elemento miDiv");
console.log("En el primer P el HTML asociado es "+losP[0].innerHTML);
```

1.3.3.- *getElementsByName (nombre)*

Esta función devuelve un array con todos los elementos DOM del subárbol cuyo atributo name sea el indicado en la cadena "nombre".

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

Ejemplo:

```
let elementos = document.getElementsByName("name");
let i;
// Todos los textbox que tengan de name alumnos, los marcamos
for (i = 0; i < elementos.length; i++) {
    if (elementos[i].type === "checkbox") {
        elementos[i].checked = true;
    }
}
```

Desarrollo Web en Entorno Cliente

```
}  
}
```

1.4.- Funciones para crear/eliminar nodos

1.4.1.- *removeChild(nodo)*

Esta función se aplica a un nodo padre. La función recibe un nodo hijo suyo y lo borra. Es útil usarlo con el atributo “parentnode”, que devuelve el nodo padre del elemento que estamos manejando.

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

Ejemplo:

```
let parrafo=document.getElementById("miParrafo");  
// Obtiene la referencia del padre, y al padre le aplica la función removeChild  
parrafo.parentNode.removeChild(parrafo);
```

1.4.2.- *appendChild(nodo)*

Esta función se aplica a un nodo padre. La función recibe un nodo y lo incluye como nodo hijo del padre. Se puede combinar con funciones como “createElement”, que permiten crear elementos XHTML.

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

Ejemplo:

```
// Creo un nodo de tipo LI  
let nuevoNodo = document.createElement("LI");  
// Al nodo LI le asocio un texto (también podría asociarse XHTML con innerHTML)  
let nodoTexto = document.createTextNode("Agua");  
nuevoNodo.appendChild(nodoTexto);  
// A miLista, lista ya existente, le añado el elemento creado  
document.getElementById("miLista").appendChild(nuevoNodo);
```

Desarrollo Web en Entorno Cliente

2.- BROWSER OBJECT MODEL (BOM)

2.1.- Introducción

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.

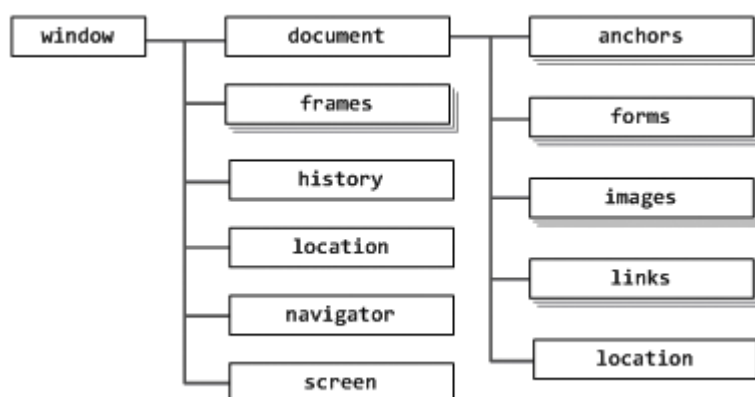
Mediante BOM, es posible redimensionar y mover la ventana del navegador, modificar el texto que se muestra en la barra de estado y realizar muchas otras manipulaciones no relacionadas con el contenido de la página HTML.

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.

Desarrollo Web en Entorno Cliente

2.2.- 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.

A continuación, se muestran algunos ejemplos de uso de estas funciones:

```
// 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 ventana  
window.moveTo(0, 0);
```

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:

Desarrollo Web en Entorno Cliente

- 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`.

2.3.- 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:

```
setTimeout("console.log('Han transcurrido 3 segundos desde que me programaron')", 3000);
```

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);
```

Como es habitual, cuando se indica la referencia a la función no se incluyen los paréntesis, ya que de otro modo, se ejecuta la función en el mismo instante en que se establece el intervalo de ejecución.

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:

Desarrollo Web en Entorno Cliente

```
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 pendiente  
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()`:

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

```
setInterval(muestraMensaje, 1000);
```

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 intervalo  
clearInterval(id);
```

2.4.- 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
<code>lastModified</code>	La fecha de la última modificación de la página
<code>referrer</code>	La URL desde la que se accedió a la página (es decir, la página anterior en el array <code>history</code>)
<code>title</code>	El texto de la etiqueta <code><title></code>
<code>URL</code>	La URL de la página actual del navegador

Desarrollo Web en Entorno Cliente

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
<code>anchors</code>	Contiene todas las "anclas" de la página (los enlaces de tipo <code></code>)
<code>applets</code>	Contiene todos los applets de la página
<code>embeds</code>	Contiene todos los objetos embebidos en la página mediante la etiqueta <code><embed></code>
<code>forms</code>	Contiene todos los formularios de la página
<code>images</code>	Contiene todas las imágenes de la página
<code>links</code>	Contiene todos los enlaces de la página (los elementos de tipo <code></code>)

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>
    
    <form method="post" name="consultas">
      <input type="text" name="id" />
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

Desarrollo Web en Entorno Cliente

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 `document.images[0].src`.

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

Desarrollo Web en Entorno Cliente

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.

Además de las propiedades de la tabla anterior, el objeto `location` contiene numerosos métodos y funciones. Algunos de los métodos más útiles son los siguientes:

```
// Método assign()
location.assign("http://www.ejemplo.com"); // Equivalente a location.href = "http://www.ejem
plo.com"

// Método replace()
location.replace("http://www.ejemplo.com");
// Similar a assign(), salvo que se borra la página actual del array history del navegador

// Método reload()
location.reload(true);
/* Recarga la página. Si el argumento es true, se carga la página desde el servidor. Si es fa
lse, se carga desde la cache del navegador */
```

2.6.- 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:

Desarrollo Web en Entorno Cliente

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

Desarrollo Web en Entorno Cliente

2.7.- 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
<code>availHeight</code>	Altura de pantalla disponible para las ventanas
<code>availWidth</code>	Anchura de pantalla disponible para las ventanas
<code>colorDepth</code>	Profundidad de color de la pantalla (32 bits normalmente)
<code>height</code>	Altura total de la pantalla en píxel
<code>width</code>	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);
```