TEMA 7. Modelo de objetos del documento (DOM).

Desarrollo Web en Entorno Cliente.

Profesor: Juan José Gallego García

Índice :

- ¿Qué es el DOM?
- Objetos del BOM
- Objetos del DOM
- Bibliografía.

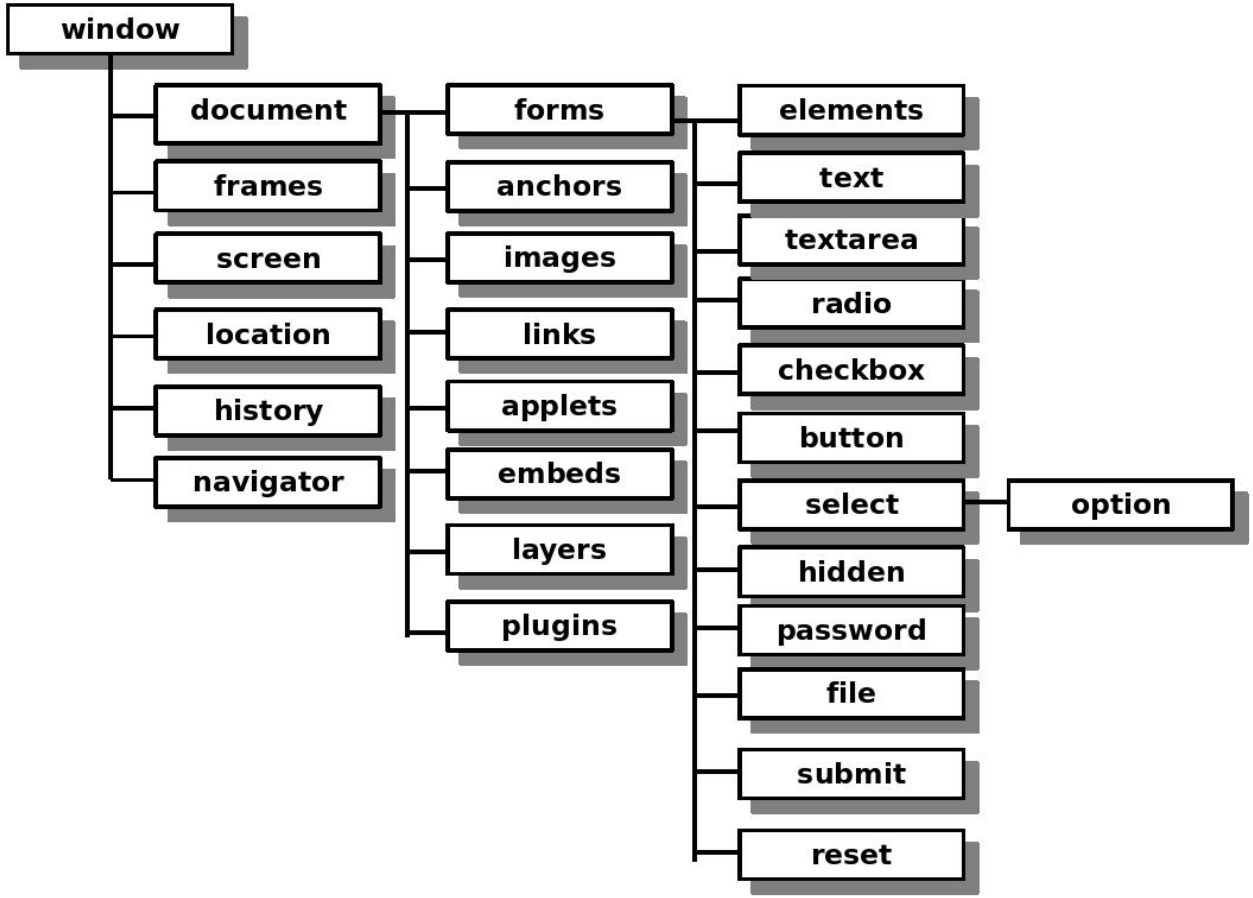
¿ Qué es el DOM?

El DOM o Document Object Model (Modelo de Objetos del Documento) es una representación jerárquica de los objetos que tiene una determinada página web creando una estructura accesible desde los lenguajes de programación del cliente, por ejemplo desde JS.

Pero además de estos objetos del documento existen otros por encima o al mismo nivel en la jerarquía que se suele confundir como objetos del DOM. Estos objetos pueden ser específicos o comunes al navegador, por ejemplo el objeto window, que en realidad es contenedor del propio DOM siendo éste último una propiedad del primero (window.document…).

A estos objetos propios del navegador se le llama BOM o Browser Object Model (Modelo de Objetos del Navegador) .

Jerarquía del BOM y DOM



Objetos del BOM

- Objeto window :

Soportado por la mayoría de los navegadores, representa la ventana del navegador está arriba en la jerarquía de los objetos que definen una web, todas las variables o funciones globales, DOM, métodos y propiedades (propias) forman parte del objeto ventana.

Por tanto todos los objetos llevan como prefijo window aunque generalmente se puede omitir:

equivalentes

```
window.document.getElementById("demo").innerHTML
```

```
document.getElementById("demo").innerHTML
```

Algunas propiedades y métodos de window :

```
window.innerHeight - devuelve la altura(px)interior del navegador
window.innerWidth - devuelve la anchura(px)interior del navegador
window.open() - abrir una nueva ventana
window.close() - cerrar la ventana actual
window.moveTo() - mover la ventana actual
window.resizeTo() - cambiar el tamaño de la ventana actual
```

Otras propiedades y métodos

- Objeto screen :

window.screen o screen, contiene información de la pantalla del usuario.

Propiedades :

```
screen.width // Devuelve ancho de la pantalla en px.
screen.height // Devuelve alto de la pantalla en px.
screen.availWidth // Devuelve ancho de la pantalla restando barra de windows (p.e.).
screen.availHeight // Devuelve alto de la pantalla restando barra de windows (p.e.).
screen.colorDepth // Devuelve profundidad de color de la pantalla (normalmente 24 bits)
```

Ejemplo:

```
<body>
<p id="p1"></p>
<script>
    document.getElementById("p1").innerHTML = "Ancho de la pantalla " + screen.width + " px";
</script>
</body>
```

- Objeto location :

Contiene información sobre la URL actual y puede redirigir el navegador a una nueva página.

Propiedades y métodos:

```
location.href      // devuelve la URL de la página actual.
location.hostname  // devuelve el nombre de dominio del proveedor de alojamiento web.
location.pathname  // devuelve la ruta y el nombre de la página actual.
location.protocol  // devuelve el protocolo web utilizado (http: o https :).
location.search    // devuelve los parámetros de la URL.
location.assign()  // carga un nuevo documento (URL).
location.reload()  // recarga la página actual.
```

Ejemplo:

```
function redirec () {
  location.assign("https://ieslosremedios.org");
}
...
<p id="p1" onclick="redirec();">Ir a IESLosRemedios</p>
```

- Objeto history :

Contiene propiedades y métodos para devolver información sobre el historial del navegador.

Propiedades y métodos:

```
history.length; // devuelve el número de URLs que contiene el historial actualmente
history.back()      // equivalente al botón retroceder URL del navegador.
history.forward()   // equivalente al botón avanzar URL del navegador.
history.go()       //carga una URL específica del historial del navegador a partir de
                  // un parámetro negativo (hacia atrás) y positivo (hacia adelante)
```

Ejemplos equivalentes:

```
function atrasURL () {
   history.back();
}...
<p id="p1" onclick="atrasURL();">Atrás</p>
```

```
function atrasURL () {
   history.go(-1);
}...
<p id="p1" onclick="atrasURL();">Atrás</p>
```

- Objeto navigator :

Contiene información sobre el navegador del usuario a través de una API.

Propiedades y métodos más destacables:

```
navigator.appVersion    // devuelve información sobre la versión del navegador.
navigator.cookieEnabled // devuelve TRUE si los cookies están habilitados.
navigator.geolocation   // devuelve un objeto usado para geolocalizar al usuario.
navigator.language  // devuelve el lenguaje del navegador.
navigator.javaEnabled() // devuelve TRUE si el navegador tiene habilitado Java.
```

navigator.geolocation

Propiedades y métodos

```
coords      // devuelve latitud y longitud de la posición del dispositivo.
position    // devuelve información de posición como objeto y pasado por parámetro.
positionError     // devuelve error de posición.
getCurrentPosition() // devuelve la posición actual geolocalizada.
watchPosition() // realiza un seguimiento del dispositivo en movimiento calculando la posición.
clearWatch()  // deja de hacer seguimiento cuando el dispositivo está en movimiento.
```

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.    <script>
5.      function getLocation() {
6.        if (navigator.geolocation) {
7.          navigator.geolocation.getCurrentPosition(showPosition);
8.        } else {
9.          alert( "El navegador no soporta geolocalización.");
10.       }
11.     }
12.     function showPosition(position) {
13.      alert ("Latitud: " + position.coords.latitude +
14.       "\nLongitud: " + position.coords.longitude);
15.     }
16.     </script>
17. </head>
18. <body>
19. <p>Clic para ver tus coordenadas.</p>
20. <button onclick="getLocation()">Prueba</button>
21. </body>
22. </html>
```

- Objetos alert( ), confirm( ) y prompt( ) :

Ya conocemos estos objetos y se pueden referenciar con o sin window, por ejemplo window.alert( ) ó alert ( ).

- Objeto console :

Referenciado como window.console(texto) ó console(texto) ,proporciona acceso a la ventana de depuración del navegador.

Métodos más destacables:

```
console.clear()      // limpia los mensajes de consola.
console.log()        // envía un mensaje a la consola.
console.table()      // muestra la información en forma de tabla.
console.time()       // comienza contador (para calcular tiempo de operaciones)
console.timeEnd()    // para contador iniciado con console.time()
console.trace()      // envía mensajes de seguimiento en llamadas a funciones.
```

Ejemplo --->

```
<script>
    console.log("Prueba de tiempo de operación");
    console.time("test1"); // comienza contador para test1 en ms.
    var alumnos=["Juan","Pepe","Pedro"];
    console.table(alumnos); // Salida de tabla
    console.timeEnd("test1"); // devuelve tiempo del test1 en ms.
</script>
```

Salida consola :

Prueba de tiempo de operación — p2.html:6

p2.html:9

(index)	Value
0	"Juan"
1	"Pepe"
2	"Pedro"

▶ Array(3)

test1: 0.18798828125ms — p2.html:10

- Métodos setTimeout( ), setInterval( ), clearTimeout( ),clearInterval( ) :

Son métodos del objeto window para retardar (en ms.) la ejecución de una función, o que la función se ejecute repetidamente cada cierto tiempo.

Sintaxis :

- setTimeout(función , milisegundos) *// Retarda la ejecución de una función.*
- clearTimeout (nombre_variab_set) // Para la ejecución establecida con setTimeout ( )
- setInterval (función, milisegundos) *// Ejecuta una función cada intervalo de tiempo establecido.*
- clearInterval (nombre_variab_set) // Para la ejecución establecida con setInterval ( )

Ejemplo:

```
<script>
  function muestra() {
      alert("Hola");
  }
</script
```

```
<button onclick="inicia = setTimeout(muestra, 2000)">Iniciar</button>
<button onclick="clearTimeout(inicia)">Parar</button>
```

Ejemplo:

```
<div id="hora"></div>
<button onclick="inicia1=inicia()">Iniciar</button>
<button onclick="clearInterval(inicia1)">Parar</button>
```

Muestra la hora cada segundo ….

```
<script>
function inicia() {
    var on = setInterval(reloj, 1000);
    return on;
}
function reloj() {
    var actual = new Date();
    var h = actual.getHours();
    var m = actual.getMinutes();
    var s = actual.getSeconds();
    m = necesita0(m);
    s = necesita0(s);
    document.getElementById("hora").innerHTML =
      h + ":" + m + ":" + s;
}
function necesita0(i) {
    if (i < 10) {
        i = "0" + i;
    }
    return i;
}
</script>
```

- # Cookies

Los cookies son pequeños fragmentos de información almacenadas en el navegador para guardar datos sobre las visitas realizadas en páginas web y así poder consultar la actividad anterior.

- Se pueden crear, leer, modificar y borrar cookies con JS.
- Las cookies se guardan como par de clave-valor.
- Las cookies pueden tener caducidad (según la fecha y hora establecida)
- También se le puede asociar a una URL específica pero por defecto pertenece a la URL actual.

Sintaxis : *document.cookie="Clave=dato; expires= …. ; path=/ ..."*

Ejemplo →

Ejemplo: (por seguridad, algunos navegadores no permiten crear cookies sin nombre de dominio).

```
<script>
    var d = new Date();
    d.setTime(d.getTime() + (1*24*60*60*1000)); // Hora actual + 1 día de duración.
    var expira = "expires="+ d.toUTCString();
    document.cookie="nombre=Juan"+ ";" +expira + ";path=/"; // Crea cookie.
    alert(document.cookie); // Muestra cookie.
    document.cookie="nombre=Juan"+ ";" +"max-age=0;"+";path=/"; // Borra cookie.
    alert(document.cookie); // Muestra cookie borrada.
</script>
```

Ejecutar ejemplo...

Más información sobre los parámetros admitidos: cookies

- Objeto de almacenamiento web localStorage y sessionStorage

Son objetos de window y permiten almacenar indefinidamente (localStorage) un dato (clave-valor) o hasta que se cierra la pestaña (sessionStorage).

Propiedades y métodos:

```
key(n)                // Devuelve el nombre de clave almacenada especificado por 'n' (índice).
length                // Devuelve el número de datos almacenados en el objeto.
getItem(keyname)      // Devuelve el valor de la clave especificada.
setItem(keyname, value) // Añade o modifica el valor de la clave especificada.
removeItem(keyname) // Borra la clave especificada en el objeto de almacenamiento.
clear()               // Borra todas las claves de datos en el objeto de almacenamiento.
```

Ejemplo :

```
<script>
    localStorage.setItem("nombre", "Juan");
    alert(localStorage.getItem("nombre"));
</script>
```

Objetos del DOM

- Objeto document

Perteneciente al objeto window, es el nodo raíz del documento HTML. Puede ser referenciado sin el prefijo window. Además de sus propiedades y métodos contiene referencia a colecciones o elementos HTML.

Sintaxis : ***document.propiedad_método_elementoHtml ...***

Propiedades y métodos más destacables:

Las siguientes URLs contienen un listado completo de propiedades y métodos aplicable a document, y a cualquier elemento HTML, no obstante se mostrarán algunos ejemplos con lo más destacable.

Propiedades y métodos (document) Propiedades y métodos (elementos HTML)

Ejemplos:

- El siguiente ejemplo aplica una clase definida a un elemento H1. ejemplo1

- Devuelve el nombre de etiqueta Html del elemento que tiene el foco. ejemplo2
- Muestra el número de atributos, nombre y valor del segundo atributo del elemento botón. ejemplo3
- Obtiene y muestra la URL del documento. ejemplo4
- Muestra el uso de write ( ) . ejemplo5
- Resumen de selección de elementos por Id, nombre de etiqueta y nombre del elemento. ejemplo6
- Selecciona los elementos a través de una consulta y un determinado criterio, por ejemplo que contengan una clase, que sean elementos HTML de un determinado tipo, etc., a continuación cambia sus estilos. ejemplo7
  Listado de propiedades del objeto style del DOM

- Crea elementos HTML y atributos, aplica estilos, borra estilos y borra elementos. ejemplo8

Bibliografía

- LibrosWeb
- W3Schools
- Developer Mozilla Docs