Introducción a JavaScript

Francisco Javier Arce Anguiano

10: Objetos del navegador BOM (Browser Object Model)

Objetivo: El alumno manejara los objetos propios del navegador con JavaScript.

10.1. Objetos Navegador

En este capítulo, veremos de manera superficial las características más relevantes de los objetos del navegador JavaScript.

Cuando se carga una página en un navegador, se crea un número de objetos característicos del navegador según el contenido de dicha página. A continuación veremos los objetos y propiedades que tiene un documento:

- window. Es el objeto de más alto nivel, contiene las propiedades de la ventana y en el supuesto de trabajar con frames, un objeto window es generado para cada frame.
- location. Contiene las propiedades de la URL activa.
- history. Objeto que contiene las propiedades que representan a las URL que el usuario ha visitado anteriormente. Es una especia de caché.
- document. Este objeto contiene todas las propiedades del documento actual, como pueden ser su color de fondo, enlaces, imágenes, etc.

Los objetos en el navegador se rigen por una jerarquía que refleja la estructura de los documentos HTML. Según esto, el objeto window que es el de más alto nivel, tendría a un objeto location como descendiente.

Imaginemos un documento HTML(object document) que contiene un formulario llamado miformulario. Para hacer referencia al formulario se deberá escribir:

```
document.miformulario
```

Como norma general para referenciar una propiedad específica de un objeto para referenciar una propiedad específica de un objeto se deberá incluir el objeto y todos

sus antecesores teniendo en cuenta que el objeto window no es necesario incluirlo a no se que se esté trabajado con frames. Veamos a continuación la jerarquía de los objetos de un navegador.

window

- parent, frames, self, top
- history
- location
- document
    - links
    - anchor
    - form
        - Todos sus elementos

10.2. El objeto Window

El objeto window posee una serie de propiedades que determinan características básicas de la ventana y sus componentes. A continuación las propiedades mas elementales:

- closed. Propiedad que determina si una ventana se ha cerrado.
- defaultStatus. Propiedad que contiene el mensaje estándar que aparece en la barra de estado de windows.
- frames. Es una matriz que representa todos los frames de la ventana.
- lenght. Esta propiedad contiene el número de frames de la ventana.
- name. Contiene el nombre de la ventana.
- outerHeight. Altura de la totalidad de la ventana.
- outerWidth. La anchura de la totalidad de la ventana.
- parent. Hace referencia a la ventana con un código <FRAMESET>/
- self. Propiedad que hace referencia a la ventana activa.
- top. Hace referencia a la ventana superior del navegador.
- status. Determina el mensaje que aparece en la barra de estado del navegador.
- window. Al igual que self, hace referencia a la ventana activa.

Ejemplo: Diseñaremos un programa que muestre siempre un texto en la barra de estado y que oculte la dirección real de un enlace al pasar el cursor del ratón sobre el.

```
<body onload="window.defaultStatus='Curso basico de JavaScript';">
<a href="http://www.enlace.com/"
onMouseOver="window.status='Estas encima del enlace';return true">
pasa por encima mio.</A>
</body>
```

Ejemplo de aplicación de los métodos alert(), confirm() y prompt():

```
<html>
<head>
<title>Ejemplo de ABRIR y CERRAR una ventana</title>
<script language="javascript">

function Pregunta(){
var EntradaDatosPregunta=prompt("Intoduce tu nombre, por favor","en
minusculas,gracias");
   if(confirm("Estas conforme con el nombre
introducido"+EntradaDatosPregunta+"?"))
   {
       alert("De acuerdo, escribiste"+EntradaDatosPregunta);
   } else {
      alert("Bueno, pero yo creo que
escribiste"+EntradaDatosPregunta);
   }
}
 </script>
 </head>
<body>
<form>
<input type="button" value="Pulsa Aqui" onClick="Pregunta()">
</form>
</body>
```

```
</html>
```

10.3. Crear ventanas emergentes con los objetos *window* y *document*

El objeto window también posee una serie de métodos que permiten ejecutar funciones específicas con las ventanas, como por ejemplo crear ventanas y cuadros de diálogo.

- open() y close(). Métodos que abren y cierran una ventana.
- back(). Retrocede a la página anterior.
- blur(). Quita el foco de la ventana especificada.
- captureEvents(). Captura todos los eventos de un determinado tipo.
- clearInterval(). Cancela el tiempo de espera establecido mediante setInterval().
- close(). Cierra la ventana.
- alert(). Método que muestra una ventana de diálogo con un mensaje y el botón Aceptar.
- confirm(). Método similar al anterior, pero mostrando dos botones, Aceptar y Cancelar.
- find(). Abre una ventana de diálogo que permite efectuar búsquedas.
- prompt(). Método que representa una ventana de diálogo con mensaje y un campo de entrada.
- setTimeout. Este método retrasa la ejecución de una instrucción.
- clearTimeout. Método que permite anular el timeout fijado con el método anterior.

También podemos determinar el aspecto que tendría la nueva ventana del navegador mediante una serie de componentes que permiten configurar el menú, la barra de herramientas, la barra de estado, etc. Las opciones son:

- toolbar. Muestra la barra de herramientas.
- location. Muestra la barra de dirección.
- directories. Muestra los botones de directorio.
- status. Muestra la barra de estado.
- menubar. Muestra la barra de menús.
- scrollbars. Muestra las barras de desplazamiento.

- resizable. Permite ajustar el tamaño de la ventana.
- width. Ancho de la ventana en pixeles.
- height. Altura de la ventana en pixeles.

Para abrir una ventana utilizando los métodos y opciones anteriores, deberemos aplicar la siguiente sintaxis:

```
variableVentana=
nombreventana.open("URL","NombreVentana","OpcionesVentana");
```

Script que abre una ventana nueva cuando se pulsa un botón.

```
<html>
<head>
<title>Ejemplo de ABRIR y CERRAR una ventana</title>
<script language="javascript">
function AbrirVentana()
{
Ventana=open("","nueva","toolbar=no,directories=no,menubar=no,width=1
80,height=160");
Ventana.document.wrtie("<head><title>Ventana Nueva</title>
</head><body>");
Ventana.document.write("<font size=4 COLOR=blue>VENTANA
NUEVA</font><br>");
Ventana.document.write("<form><input type='button' VALUE =
'Cerrar'onClick='self.close()'> </form>");
}
 </script>
 </head>
<body>
<form>
<input type="button" value="Abrir una ventana"
onClick="AbrirVentana();"><br>
</form>
</body>
```

```
</html>
```

10.4. Las propiedades innerWidhth e innerHeight

```
<!DOCTYPE html>
<html>
<head>
      <title>Bom | Size</title>
      <meta charset="utf-8">
      <script>
      window.onload = function(){
            var w = window.innerWidth ||
document.documentElement.clientWidth || document.body.clientWidth;
            var h = window.innerHeight ||
document.documentElement.clientHeight || document.body.clientHeight;
            document.getElementById("salida").innerHTML = "Dimensiones
del viewport son "+w+" ancho y "+h+" altura en pixeles y las
dimensiones externas son "+window.outerWidth+" de ancho y
"+window.outerHeight+" de altura en piexeles";
      }
       </script>
 </head>
<body>
<div id="salida"></div>
</body>
</html>
```

10.5. BOM: el objeto screen

```
document.write("Ancho: "+window.screen.width+" pixeles<br>");
document.write("Altura: "+window.screen.height+" pixeles<br>");
document.write("Ancho disponible: "+window.screen.availWidth+"
pixeles<br>");
document.write("Altura disponible: "+window.screen.availHeight+"
pixeles<br>");
document.write("Profundidad de color: "+window.screen.colorDepth+"
```

```
pixeles<br>");
document.write("Profundidad de pixel: "+window.screen.pixelDepth+"
pixeles<br>");
```

10.6. BOM: El objeto location para manejar la URL de la página actual

La propiedad location del objeto window contiene información sobre el URL completo de un documento actual a diferencia de la propiedad location del objeto document que se encarga de cargar un nuevo documento.

```
window.location.propiedad
```

Propiedades del objeto location.

- protocol. Especifica el inicio de la dirección.
- hash. Especifica el nombre del enlace en la URL.
- host. Determina el nombre del servidor y el puerto.
- href. Especifica la dirección completa.
- port. Especifica el puerto de comunicaciones.

```
<!DOCTYPE html>
<html>
<head>
     <title>BOM | Location</title>
     <meta charset="utf-8">
     <script>
          document.write("URL: "+window.location.href+"<br>");
          document.write("Hostname: "+location.hostname+"<br>");
          document.write("Ruta: "+location.pathname+"<br>");
          document.write("Protocolo: "+location.protocol+"<br>");
          document.write("Puerto: "+location.port+"<br>");
          function salta(){
               location.assign("http://www.google.com");
          }
      </script>
 </head>
```

```
<body>
     <input type="button" value="Ir a google" onclick="salta()" />
</body>
</html>
```

10.7. BOM: El objeto History

Este objeto contiene información sobre los enlaces que el usuario ha visitado. Su utilidad más aparente es la de generar botones de avance y retroceso.

- back(). Carga el URL anterior al actual.
- forward(). Carga el URL siguiente de la lista.
- go(). Muestra un URL de la lista history según un valor índice introducido.

```
<!DOCTYPE html>
<html>
<head>
     <title>BOM | History</title>
     <meta charset="utf-8">
     <script>
          function atras(){
               window.history.back();
          }
          function adelante(){
               window.history.forward();
          }
      </script>
 </head>
<body>
<input type="button" value="Atrás" onclick="atras()">
<input type="button" value="Adelante" onclick="adelante()">
</body>
</html>
```

10.8. BOM: El objeto navigator

```
document.write("Galletas activas: "+navigator.cookieEnabled+"<br>");
document.write("La app del navegador: "+navigator.appName+"<br>");
document.write("La app code name: "+navigator.appCodeName+"<br>");
document.write("El motor del navegador: "+navigator.product+"<br>");
document.write("Versión del navegador:
"+navigator.appVersion+"<br>");
document.write("User Agent: "+navigator.userAgent+"<br>");
document.write("Plataforma: "+navigator.platform+"<br>");
document.write("Lenguaje: "+navigator.language+"<br>");
document.write("En linea: "+navigator.onLine+"<br>");
document.write("Java activo: "+navigator.javaEnabled())
```

10.9. BOM: manejo del tiempo con setTimeout() y setInterval()

Manejo del reloj de la computadora con setinterval()

```
<!DOCTYPE html>
<html>
<head>
      <title>BOM | Manejo de tiempo</title>
      <meta charset="utf-8">
      <script>
      var llave = setInterval(reloj,1000);
      function reloj(){
            var d = new Date();
            document.getElementById("salida").innerHTML =
d.toLocaleTimeString();
      }
       </script>
 </head>
<body>
<p id="salida"></p>
<button onclick="clearInterval(llave);">Detener</button>
</body>
</html>
```

Manejo de tiempo son setTimeout()

```
<!DOCTYPE html>
<html>
<head>
	<title>BOM | Manejo de tiempo</title>
	<meta charset="utf-8">
	<script>
	function iniciar(){
		alert("Hola desde JavaScript");
	}
	 </script>
 </head>
<body>
<input type="button" value="Iniciar" onclick="llave =
setTimeout(iniciar,3000);">
<input type="button" value="Detener" onclick="clearTimeout(llave);">
</body>
</html>
```

10.10. BOM: Crear y leer galletas o cookies en JavaScript

```
<!DOCTYPE html>
<html>
<head>
	<title>BOM | cookies</title>
	<meta charset="utf-8">
	<script>
	function leeGalleta(galleta){
		var usuario = galleta +"=";
		var galletaLimpia = decodeURIComponent(document.cookie);
		var ca = galletaLimpia.split(";");
		for (var i = 0; i < ca.length; i++) {
			c = ca[i];
			while(c.charAt(0)==" "){
				c = c.substring(1);
			}
			if(c.indexOf(usuario)==0){
				return c.substring(usuario.length, c.length);
```

```
                }
                return "";
            }
        }
        function guardaGalleta(nombreGalleta, valorGalleta, dias){
            console.log(nombreGalleta, valorGalleta, dias);
            var d = new Date();
            d.setTime(d.getTime()+(dias*24*60*60*1000));
            var expira = "expires="+d.toGMTString();
            var galleta =
nombreGalleta+"="+valorGalleta+";"+expira+";path=/";
            console.log(galleta);
            document.cookie = galleta;
        }
        window.onload = function(){
            var usuario = leeGalleta("usuario");
            if (usuario=="") {
                usuario = prompt("¿Cuál es tu nombre?");
                if (usuario!="" && usuario!=null) {
                    guardaGalleta("usuario",usuario,3);
                }
            } else {
                alert("Bienvenido "+usuario+" a nuestra página");
            }
        }
        </script>
    </head>
<body>
</body>
</html>
```

10.11. El objeto document

El objeto *document* hace referencia a todo el contenido de un documento HTML.

Todas las propiedades de este objeto hacen referencia a determinadas características de la página, como su color de fondo, su título, etc. A continuación se relacionan algunas de las más utilizadas:

- bgColor. Color del fondo.
- fgColor. Color del texto.
- vlinkColor. Color de los enlaces visitados.
- alinkColor. Color del enlace en el momento de la selección.

También podemos trabajar con algunos de sus métodos para controlar el proceso de abrir y cerrar un documento.

- clear(). Borra la página del navegador.
- close(). Cierra el documento.
- write(). Permite escribir en un documento.