Lenguajes de script

Un lenguaje de *script* es un pequeño lenguaje de programación cuyo código se inserta dentro del documento HTML. Este código se ejecuta en el navegador del usuario al cargar la página, o cuando sucede algo especial como puede ser el pulsar sobre un enlace.
Estos lenguajes permiten variar dinámicamente el contenido del documento, modificar el comportamiento normal del navegador, validar formularios, realizar pequeños trucos visuales, etc... Sin embargo, conviene recordar que se ejecutan en el navegador del usuario y no en la máquina donde estén alojadas, por lo que no podrán realizar cosas como manejar bases de datos. Esto hace que los contadores (por ejemplo) se deban realizar de otra manera, utilizando programas CGI.
El primer lenguaje de *script* que vió la luz fue el JavaScript de Netscape. Nacido con la versión 2.0 de este navegador y basado en la sintaxis de Java, su utilidad y el casi absoluto monopolio que entonces ejercía Netscape en el mercado de navegadores permitieron que se popularizara y extendiera su uso.
El máximo rival del Netscape Navigator, el Internet Explorer de Microsoft, comenzó a soportar este lenguaje en su versión 3.0. Fue también entonces cuando introdujo el único rival serio que el JavaScript ha tenido en el mercado de los lenguajes de *script*: el VBScript. Basado en el lenguaje BASIC, no ha tenido excesiva difusión en Internet debido a la previa implantación del JavaScript y a que son de parecida funcionalidad, pero sí es utilizado dentro de Intranets basadas en el Explorer y dentro de otras aplicaciones de Microsoft, como IIS, Access, Word, etc.

Javascript
Por el momento y dado que pronto veremos PHP no vamos a ver nada de VBScript. Pero para ilustrar la utilidad de los lenguajes de *script*, vamos a realizar una pequeña introducción al Javascript.
Vamos a realizar nuestro primer "programa" en JavaScript. Haremos surgir una ventana que nos muestre el famoso mensaje "hola, mundo". Así podremos ver los elementos principales del lenguaje. El siguiente código es una página Web completa con un botón que, al pulsarlo, muestra el mensaje.

```
<HTML>
<HEAD>
 <SCRIPT LANGUAGE="JavaScript">
 <!---
   function HolaMundo() {
    alert("¡Hola, mundo!");
   }
 // --->
 </SCRIPT>
</HEAD>
<BODY>
<FORM>
        <INPUT TYPE="button" NAME="Boton" VALUE="Pulsame"
onClick="HolaMundo()">
</FORM>
</BODY>
</HTML>
```

Guarda es te archivo con extensión htm o html y ejecútalo.

Ahora vamos a ver, paso por paso, que significa cada uno de los elementos extraños que tiene la página anterior:

```
<SCRIPT LANGUAGE="JavaScript">
</SCRIPT>
```

Dentro de estos elementos será donde se puedan poner funciones en JavaScript. Puedes poner cuantos quieras a lo largo del documento y en el lugar que más te guste. Si un navegador no entiende la etiqueta <SCRIPT> escribirá lo que hay entre medias de estos elementos, así que lo encerramos entre comentarios por si las moscas.

```
 function HolaMundo() {
   alert("¡Hola, mundo!");
 }
```

Esta es nuestra primera función en JavaScript. En el código de la misma vemos una llamada al método alert (que pertenece al objeto window) que es la que se encarga de mostrar el mensaje en pantalla. Por un fallo del Netscape no se pueden poner las etiquetas HTML de caracteres especiales en una función: no los reconoce. Así que pondremos directamente "¡" arriesgándonos a que salga de otra manera en ordenadores con un juego de caracteres distinto al del nuestro.

```
<FORM>
        <INPUT TYPE="button" NAME="Boton" VALUE="Pulsame"
onClick="HolaMundo()">
</FORM>
```

Dentro del elemento que usamos para mostrar un botón vemos una cosa nueva: onClick. Es un *evento*. Cuando el usuario pulsa el botón, el evento onClick se dispara y ejecuta el código que tenga entre comillas, en este caso la llamada a la función HolaMundo(), que tendremos que haber definido con anterioridad.
Este ejemplo muestra una pequeña parte de las funcionalidades del JavaScript.

Elementos básicos de JavaScript

De nuevo comenzamos por lo más sencillo. Pero, ojo, que este lenguaje es más complicado que el VBScript, aunque también más potente.

Comentarios

En JavaScript existen dos tipos de comentarios. El primero es equivalente al de VBScript y nos permite que el resto de la línea sea un comentario. Para ello se utilizan dos barras inclinadas:

```
var i = 1; // Aqui esta el comentario
```

Sin embargo, también permite un tipo de comentario que puede tener las líneas que queramos. Estos comentario comienzan con /* y terminan por */. Por ejemplo:

```
/* Aqui comienza nuestro maravilloso comentario
  que sigue por aquí
  e indefinidamente hasta que le indiquemos el final */
```

Literales

En Javascript existen más tipos de literales que en VBScript. Aparte de los distintos tipos de números y valores booleanos (true y false), también podemos especificar vectores:

```
vacaciones = ["Navidad", "Semana Santa", "Verano"];
alert(vacaciones[0]);
```

Dentro de las cadenas podemos indicar varios caracteres especiales, con significados especiales. Estos son los más usados:

Carácter	Significado
\n	Nueva línea
\t	Tabulador
\'	Comilla simple
\"	Comilla doble
\\	Barra invertida
\999	El número ASCII (según la codificación Latin-1) del carácter
\x99	El número ASCII (según la codificación Latin-1) del carácter en hexadecimal

De este modo, el siguiente literal:

```
"El curso de Javascript (\xA9 1997-99 Daniel Rodríguez) es \"co..\"."
```

se corresponde con la cadena:

```
El curso de Javascript (© 1997-99 Daniel Rodríguez) es "co..".
```

Por último, también se pueden especificar objetos como literales, aunque no funcione en más que en Netscape 4 y superiores:

```
miNavegador = {nombre: "Netscape", version: 4.5,
        idioma: "Español", plataforma: "PC"};
alert(miNavegador.plataforma);
```

Sentencias y bloques

En Javascript las sentencias se separan con un punto y coma, y se agrupan mediante llaves ({ y }).

Funciones en JavaScript

Las funciones son los únicos tipos de subprogramas que acepta JavaScript. Tienen la siguiente estructura:

```
function nombre(argumento1, argumento2,..., argumento n) {
  código de la funcion
}
```

Los parámetros se pasan por valor. Eso significa que si cambiamos el valor de un argumento dentro de una función, este cambio no se verá fuera:

```
function sumarUno(num) {
  num++;
}

var a = 1:
sumarUno(a);
```

En este ejemplo, a seguirá valiendo 1 después de llamar a la función. Esto tiene una excepción, que son las referencias. Cuando se cambia el valor de una referencia dentro de una función también se cambia fuera.

Para devolver un valor de retorno desde la función se utiliza la palabra reservada return:

```
function cuadrado(num) {
  return num * num;
}

a = cuadrado(2);
```

En este ejemplo, a valdrá 4.

Se pueden definir funciones con un número variable de argumentos. Para poder luego acceder a dichos parámetros dentro de la función se utiliza el vector arguments. Este ejemplo sumaría el valor de todos los parámetros:

```
function sumarArgumentos() {
  resultado = 0;
  for (i=0; i<arguments.length; i++)
    resultado += arguments[i];
  return resultado;
}
```

Funciones predefinidas

JavaScript dispone de las siguientes funciones predefinidas:

eval(cadena)

Ejecuta la expresión o sentencia contenida en la cadena que recibe como parámetros.

```
mensaje = 'Hola';
eval("alert('" + mensaje + "');");
```

Este ejemplo nos muestra una ventana con un saludo.

parseInt(cadena [, base])

Convierte en un número entero la cadena que recibe, asumiendo que está en la base indicada. Si este parámetro falta, se asume que está en base 10. Si fracasa en la conversión devolverá el valor NaN.

```
parseInt("3453");
```

Devuelve el número 3453.

parseFloat(cadena)

Convierte en un número real la cadena que recibe, devolviendo NaN si fracasa en el intento.

parseFloat("3.12.3");
Este ejemplo devuelve NaN ya que la cadena no contiene un número real válido.
isNaN(valor)
Devuelve true sólo si el argumento es NaN.
isFinite(numero)
Devuelve true si el número es un número válido y no es infinito.
Number(referencia)
String(referencia)
Convierten a número (o referencia) el objeto que se les pase como argumento.

Validar un formulario con Javascript.

Vamos realizar un ejemplo de un formulario completo para validar. Las validaciones se hacen en el propio navegador antes de enviarlo. Si hubo algún campo no relleno o con información errónea, el formulario muestra el campo que está incorrecto y solicita al usuario que lo cambie. Si todos los datos del formulario son correctos se envía el formulario.

He querido hacer un formulario sencillo, para que el ejercicio no se haga demasiado complicado. No obstante, se realizan validaciones en campos con distintos valores, para hacerlo más variado. Se comprueba un campo donde debe figurar un texto, otro donde debe introducirse un número mayor que 18 y un último con un campo select donde deben haber seleccionado un valor.

El código del formulario

El formulario con el que vamos a trabajar es el siguiente:

```
<form name="fvalida">
<table>
<tr>
   <td>Nombre: </td>
   <td><input type="text" name="nombre" size="30" maxlength="100"></td>
</tr>
<tr>
   <td>Edad: </td>
   <td><input type="text" name="edad" size="3" maxlength="2"></td>
</tr>
<tr>
   <td>Interés:</td>
   <td>
   <select name=interes>
   <option value="Elegir">Elegir
   <option value="Comercial">Contacto comercial
   <option value="Clientes">Atención al cliente
   <option value="Proveedores">Contacto de proveedores
   </select>
   </td>
</tr>
```

```
<tr>
   <td colspan="2" align="center"><input type="button" value="Enviar"
onclick="valida_envia()"></td>
</tr>
</table>
</form>
```

Es un formulario cualquiera. Los únicos puntos donde debemos prestar atención son:
El nombre del formulario, "fvalida", que utilizaremos para referirnos al él mediante Javascript.
El botón de enviar, que en lugar de ser un submit corriente, es un botón que llama a una función, que se encarga de validar el formulario y enviarlo si todo fue correcto.
Función Javascript para validar el formulario

Ahora veremos la función que hemos creado para validar el formulario. Se llama valida_envia(). Simplemente, para cada campo del formulario, comprueba que el valor introducido es correcto. Si no es correcto, muestra un mensaje de alerta, pone el foco de la aplicación en el campo que ha dado el error y abandona la función retornando el valor 0.

Si todos los campos eran correctos, la función continúa hasta el final, sin salirse, por no estar ningún campo incorrecto. Entonces ejecuta la sentencia última, que es el envío del formulario.

Veamos la función entera, aunque luego la expliquemos por partes.

```
function valida_envia(){
   //valido el nombre
   if (document.fvalida.nombre.value.length==0){
      alert("Tiene que escribir su nombre")
      document.fvalida.nombre.focus()
      return 0;
   }

   //valido la edad. tiene que ser entero mayor que 18
   edad = document.fvalida.edad.value
   edad = validarEntero(edad)
   document.fvalida.edad.value=edad
   if (edad==""){
      alert("Tiene que introducir un número entero en su edad.")
      document.fvalida.edad.focus()
      return 0;
   }else{
      if (edad<18){
         alert("Debe ser mayor de 18 años.")
         document.fvalida.edad.focus()
         return 0;
      }
   }
```

```
    //valido el interés
    if (document.fvalida.interes.selectedIndex==0){
      alert("Debe seleccionar un motivo de su contacto.")
      document.fvalida.interes.focus()
      return 0;
    }

    //el formulario se envia
    alert("Muchas gracias por enviar el formulario");
    document.fvalida.submit();
}
```

En el primer bloque se valida el campo nombre. La validación que se hace es simplemente si se ha escrito algo en el campo. Para ello comprueba si el número de caracteres escritos en el campo nombre es cero. En ese caso, muestra el mensaje de alerta, sitúa el foco en el campo de texto y se sale de la función devolviendo el valor 0.

> Nota: el foco de la aplicación es el lugar donde está situado el cursor. El foco puede estar en cualquier sitio. Por ejemplo en un campo de texto, en un select, en un enlace o en la propia página. Si presionamos una tecla del teclado afecta al lugar donde está situado el foco. Si, por ejemplo, el foco está en un campo de texto, al operar con el teclado estaremos escribiendo en ese campo de texto.

La validación de la edad mayor que 18 años tiene dos partes. Primero debemos comprobar que en el campo de texto hay escrito un valor entero. Luego, si teníamos un entero, habría que comprobar que es mayor que 18. Para hacer esta validación nos vamos a apoyar en una función que devuelve un string vació en caso de que no sea un entero y el propio entero, si es que lo era.

Antes de realizar la validación de la edad propiamente dicha, se obtiene el valor introducido en el campo de formulario edad y se guarda en una variable llamada edad. Luego se ejecuta la función pasando esta edad. Su resultado lo utilizamos para volcarlo otra vez al campo de texto. Entonces, se comprueba si el valor devuelto por la función es un string vacío. En ese caso, es que el valor escrito en el formulario no era un entero, por lo que se muestra el mensaje de error, se sitúa el foco y se sale de la función.

En caso de que el campo edad contuviese un entero, se debe comprobar a continuación si es mayor que 18. En caso de que sea menor, se muestra el error y se sale. En caso contrario -entonces el valor sería mayor o igual que 18-, se continúa con las comprobaciones.

Por último se valida el campo select, donde aparece el interés del supuesto visitante, que le motiva para enviarnos el formulario. En ese campo se debe haber seleccionado cualquier opción menos la primera. Para asegurarnos, simplemente se comprueba si el atributo selectedIndex del campo select tiene el valor 0. Ese atributo almacena el índice seleccionado en el menú desplegable. El primer campo tiene el índice 0, el segundo el índice 1...

Si se comprueba que selectedIndex vale 0, se muestra un mensaje de alerta, se pone el foco en el campo del formulario y se sale de la función.

Si hemos llegado hasta este punto sin salirnos de la función es que todos los campos del formulario estaban rellenos correctamente. En ese caso se debe enviar el formulario. Antes de enviar el formulario se muestra un mensaje de alerta, agradeciendo que se haya rellenado correctamente. Este mensaje se puede suprimir si se desea.

Para enviar el formulario se hace una llamada al método submit() de dicho formulario.

Conclusión

Este ejercicio es de lo más básico y útil que se puede hacer en Javascript. Requiere ciertos conocimientos, ya ligeramente avanzados, pero en el fondo no resulta complicado. Incluso ampliarlo es bastante sencillo, siempre que sigamos un esquema similar para cada uno de los campos.

Aquí tienen otro ejemplo:
http://www.elcodigo.net/tutoriales/jsavanzado/jsavanzado13.html