**7.2. Utilidades básicas para formularios**

**7.2.1. Obtener el valor de los campos de formulario**

La mayoría de técnicas JavaScript relacionadas con los formularios requieren leer y/o modificar el valor de los campos del formulario. Por tanto, a continuación se muestra cómo obtener el valor de los campos de formulario más utilizados.

**7.2.1.1. Cuadro de texto y textarea**

El valor del texto mostrado por estos elementos se obtiene y se establece directamente mediante la propiedad value.

<input type="text" id="texto" />

var valor = document.getElementById("texto").value;

<textarea id="parrafo"></textarea>

var valor = document.getElementById("parrafo").value;

**7.2.1.2. Radiobutton**

Cuando se dispone de un grupo de *radiobuttons*, generalmente no se quiere obtener el valor del atributo value de alguno de ellos, sino que lo importante es conocer cuál de todos los *radiobuttons* se ha seleccionado. La propiedad checked devuelve true para el *radiobutton* seleccionado y false en cualquier otro caso. Si por ejemplo se dispone del siguiente grupo de *radiobuttons*:

<input type="radio" value="si" name="pregunta" id="pregunta\_si"/> SI

<input type="radio" value="no" name="pregunta" id="pregunta\_no"/> NO

<input type="radio" value="nsnc" name="pregunta" id="pregunta\_nsnc"/> NS/NC

El siguiente código permite determinar si cada *radiobutton* ha sido seleccionado o no:

var elementos = document.getElementsByName("pregunta");

for(var i=0; i<elementos.length; i++) {

alert(" Elemento: " + elementos[i].value + "\n Seleccionado: " + elementos[i].checked);

}

**7.2.1.3. Checkbox**

Los elementos de tipo *checkbox* son muy similares a los *radiobutton*, salvo que en este caso se debe comprobar cada *checkbox* de forma independiente del resto. El motivo es que los grupos de *radiobutton*son mutuamente excluyentes y sólo se puede seleccionar uno de ellos cada vez. Por su parte, los*checkbox* se pueden seleccionar de forma independiente respecto de los demás.

Si se dispone de los siguientes *checkbox*:

<input type="checkbox" value="condiciones" name="condiciones" id="condiciones"/> He leído y acepto las condiciones

<input type="checkbox" value="privacidad" name="privacidad" id="privacidad"/> He leído la política de privacidad

Utilizando la propiedad checked, es posible comprobar si cada *checkbox* ha sido seleccionado:

var elemento = document.getElementById("condiciones");

alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);

elemento = document.getElementById("privacidad");

alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);

**7.2.1.4. Select**

Las listas desplegables (<select>) son los elementos en los que es más difícil obtener su valor. Si se dispone de una lista desplegable como la siguiente:

<select id="opciones" name="opciones">

<option value="1">Primer valor</option>

<option value="2">Segundo valor</option>

<option value="3">Tercer valor</option>

<option value="4">Cuarto valor</option>

</select>

En general, lo que se requiere es obtener el valor del atributo value de la opción (<option>) seleccionada por el usuario. Obtener este valor no es sencillo, ya que se deben realizar una serie de pasos. Además, para obtener el valor seleccionado, deben utilizarse las siguientes propiedades:

* options, es un array creado automáticamente por el navegador para cada lista desplegable y que contiene la referencia a todas las opciones de esa lista. De esta forma, la primera opción de una lista se puede obtener mediante document.getElementById("id\_de\_la\_lista").options[0].
* selectedIndex, cuando el usuario selecciona una opción, el navegador actualiza automáticamente el valor de esta propiedad, que guarda el índice de la opción seleccionada. El índice hace referencia al array options creado automáticamente por el navegador para cada lista.

// Obtener la referencia a la lista

var lista = document.getElementById("opciones");

// Obtener el índice de la opción que se ha seleccionado

var indiceSeleccionado = lista.selectedIndex;

// Con el índice y el array "options", obtener la opción seleccionada

var opcionSeleccionada = lista.options[indiceSeleccionado];

// Obtener el valor y el texto de la opción seleccionada

var textoSeleccionado = opcionSeleccionada.text;

var valorSeleccionado = opcionSeleccionada.value;

alert("Opción seleccionada: " + textoSeleccionado + "\n Valor de la opción: " + valorSeleccionado);

Como se ha visto, para obtener el valor del atributo value correspondiente a la opción seleccionada por el usuario, es necesario realizar varios pasos. No obstante, normalmente se abrevian todos los pasos necesarios en una única instrucción:

var lista = document.getElementById("opciones");

// Obtener el valor de la opción seleccionada

var valorSeleccionado = lista.options[lista.selectedIndex].value;

// Obtener el texto que muestra la opción seleccionada

var valorSeleccionado = lista.options[lista.selectedIndex].text;

Lo más importante es no confundir el valor de la propiedad selectedIndex con el valor correspondiente a la propiedad value de la opción seleccionada. En el ejemplo anterior, la primera opción tiene unvalue igual a 1. Sin embargo, si se selecciona esta opción, el valor de selectedIndex será 0, ya que es la primera opción del array options (y los arrays empiezan a contar los elementos en el número 0).

**7.2.2. Establecer el foco en un elemento**

En programación, cuando un elemento está seleccionado y se puede escribir directamente en el o se puede modificar alguna de sus propiedades, se dice que tiene el foco del programa.

Si un cuadro de texto de un formulario tiene el foco, el usuario puede escribir directamente en el sin necesidad de pinchar previamente con el ratón en el interior del cuadro. Igualmente, si una lista desplegable tiene el foco, el usuario puede seleccionar una opción directamente subiendo y bajando con las flechas del teclado.

Al pulsar repetidamente la tecla TABULADOR sobre una página web, los diferentes elementos (enlaces, imágenes, campos de formulario, etc.) van obteniendo el foco del navegador (el elemento seleccionado cada vez suele mostrar un pequeño borde punteado).

Si en una página web el formulario es el elemento más importante, como por ejemplo en una página de búsqueda o en una página con un formulario para registrarse, se considera una buena práctica de usabilidad el asignar automáticamente el foco al primer elemento del formulario cuando se carga la página.

Para asignar el foco a un elemento de XHTML, se utiliza la función focus(). El siguiente ejemplo asigna el foco a un elemento de formulario cuyo atributo id es igual a primero:

document.getElementById("primero").focus();

<form id="formulario" action="#">

<input type="text" id="primero" />

</form>

Ampliando el ejemplo anterior, se puede asignar automáticamente el foco del programa al primer elemento del primer formulario de la página, independientemente del id del formulario y de los elementos:

if(document.forms.length > 0) {

if(document.forms[0].elements.length > 0) {

document.forms[0].elements[0].focus();

}

}

El código anterior comprueba que existe al menos un formulario en la página mediante el tamaño del array forms. Si su tamaño es mayor que 0, se utiliza este primer formulario. Empleando la misma técnica, se comprueba que el formulario tenga al menos un elemento (if(document.forms[0].elements.length > 0)). En caso afirmativo, se establece el foco del navegador en el primer elemento del primer formulario (document.forms[0].elements[0].focus();).

Para que el ejemplo anterior sea completamente correcto, se debe añadir una comprobación adicional. El campo de formulario que se selecciona no debería ser de tipo hidden:

if(document.forms.length > 0) {

for(var i=0; i < document.forms[0].elements.length; i++) {

var campo = document.forms[0].elements[i];

if(campo.type != "hidden") {

campo.focus();

break;

}

}

}

**7.2.3. Evitar el envío duplicado de un formulario**

Uno de los problemas habituales con el uso de formularios web es la posibilidad de que el usuario pulse dos veces seguidas sobre el botón "Enviar". Si la conexión del usuario es demasiado lenta o la respuesta del servidor se hace esperar, el formulario original sigue mostrándose en el navegador y por ese motivo, el usuario tiene la tentación de volver a pinchar sobre el botón de "Enviar".

En la mayoría de los casos, el problema no es grave e incluso es posible controlarlo en el servidor, pero puede complicarse en formularios de aplicaciones importantes como las que implican transacciones económicas.

Por este motivo, una buena práctica en el diseño de aplicaciones web suele ser la de deshabilitar el botón de envío después de la primera pulsación. El siguiente ejemplo muestra el código necesario:

<form id="formulario" action="#">

...

<input type="button" value="Enviar" onclick="this.disabled=true; this.value=’Enviando...’; this.form.submit()" />

</form>

Cuando se pulsa sobre el botón de envío del formulario, se produce el evento onclick sobre el botón y por tanto, se ejecutan las instrucciones JavaScript contenidas en el atributo onclick:

1. En primer lugar, se deshabilita el botón mediante la instrucción this.disabled = true;. Esta es la única instrucción necesaria si sólo se quiere deshabilitar un botón.
2. A continuación, se cambia el mensaje que muestra el botón. Del original "Enviar" se pasa al más adecuado "Enviando..."
3. Por último, se envía el formulario mediante la función submit() en la siguiente instrucción:this.form.submit()

El botón del ejemplo anterior está definido mediante un botón de tipo <input type="button" />, ya que el código JavaScript mostrado no funciona correctamente con un botón de tipo <input type="submit" />. Si se utiliza un botón de tipo submit, el botón se deshabilita antes de enviar el formulario y por tanto el formulario acaba sin enviarse.

**7.2.4. Limitar el tamaño de caracteres de un textarea**

La carencia más importante de los campos de formulario de tipo textarea es la imposibilidad de limitar el máximo número de caracteres que se pueden introducir, de forma similar al atributo maxlength de los cuadros de texto normales.

JavaScript permite añadir esta característica de forma muy sencilla. En primer lugar, hay que recordar que con algunos eventos (como onkeypress, onclick y onsubmit) se puede evitar su comportamiento normal si se devuelve el valor false.

Evitar el comportamiento normal equivale a modificar completamente el comportamiento habitual del evento. Si por ejemplo se devuelve el valor false en el evento onkeypress, la tecla pulsada por el usuario no se tiene en cuenta. Si se devuelve false en el evento onclick de un elemento como un enlace, el navegador no carga la página indicada por el enlace.

Si un evento devuelve el valor true, su comportamiento es el habitual:

<textarea onkeypress="return true;"></textarea>

En el textarea del ejemplo anterior, el usuario puede escribir cualquier carácter, ya que el eventoonkeypress devuelve true y por tanto, su comportamiento es el normal y la tecla pulsada se transforma en un carácter dentro del textarea.

Sin embargo, en el siguiente ejemplo:

<textarea onkeypress="return false;"></textarea>

Como el valor devuelto por el evento onkeypress es igual a false, el navegador no ejecuta el comportamiento por defecto del evento, es decir, la tecla presionada no se transforma en ningún carácter dentro del textarea. No importa las veces que se pulsen las teclas y no importa la tecla pulsada, ese textarea no permitirá escribir ningún carácter.

Aprovechando esta característica, es sencillo limitar el número de caracteres que se pueden escribir en un elemento de tipo textarea: se comprueba si se ha llegado al máximo número de caracteres permitido y en caso afirmativo se evita el comportamiento habitual del evento y por tanto, los caracteres adicionales no se añaden al textarea:

function limita(maximoCaracteres) {

var elemento = document.getElementById("texto");

if(elemento.value.length >= maximoCaracteres ) {

return false;

}

else {

return true;

}

}

<textarea id="texto" onkeypress="return limita(100);"></textarea>

En el ejemplo anterior, con cada tecla pulsada se compara el número total de caracteres del textarea con el máximo número de caracteres permitido. Si el número de caracteres es igual o mayor que el límite, se devuelve el valor false y por tanto, se evita el comportamiento por defecto de onkeypress y la tecla no se añade.

Ejercicio 17

Mejorar el ejemplo anterior indicando en todo momento al usuario el número de caracteres que aún puede escribir. Además, se debe permitir pulsar las teclas Backspace, Supr. y las flechas horizontales cuando se haya llegado al máximo número de caracteres.

[Ver solución](http://www.librosweb.es/javascript/capitulo_11/ejercicio_17.html)

**7.2.5. Restringir los caracteres permitidos en un cuadro de texto**

En ocasiones, puede ser útil bloquear algunos caracteres determinados en un cuadro de texto. Si por ejemplo un cuadro de texto espera que se introduzca un número, puede ser interesante no permitir al usuario introducir ningún carácter que no sea numérico.

Igualmente, en algunos casos puede ser útil impedir que el usuario introduzca números en un cuadro de texto. Utilizando el evento onkeypress y unas cuantas sentencias JavaScript, el problema se resuelve fácilmente:

function permite(elEvento, permitidos) {

// Variables que definen los caracteres permitidos

var numeros = "0123456789";

var caracteres = " abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ";

var numeros\_caracteres = numeros + caracteres;

var teclas\_especiales = [8, 37, 39, 46];

// 8 = BackSpace, 46 = Supr, 37 = flecha izquierda, 39 = flecha derecha

// Seleccionar los caracteres a partir del parámetro de la función

switch(permitidos) {

case 'num':

permitidos = numeros;

break;

case 'car':

permitidos = caracteres;

break;

case 'num\_car':

permitidos = numeros\_caracteres;

break;

}

// Obtener la tecla pulsada

var evento = elEvento || window.event;

var codigoCaracter = evento.charCode || evento.keyCode;

var caracter = String.fromCharCode(codigoCaracter);

// Comprobar si la tecla pulsada es alguna de las teclas especiales

// (teclas de borrado y flechas horizontales)

var tecla\_especial = false;

for(var i in teclas\_especiales) {

if(codigoCaracter == teclas\_especiales[i]) {

tecla\_especial = true;

break;

}

}

// Comprobar si la tecla pulsada se encuentra en los caracteres permitidos

// o si es una tecla especial

return permitidos.indexOf(caracter) != -1 || tecla\_especial;

}

// Sólo números

<input type="text" id="texto" onkeypress="return permite(event, 'num')" />

// Sólo letras

<input type="text" id="texto" onkeypress="return permite(event, 'car')" />

// Sólo letras o números

<input type="text" id="texto" onkeypress="return permite(event, 'num\_car')" />

El funcionamiento del script anterior se basa en permitir o impedir el comportamiento habitual del evento onkeypress. Cuando se pulsa una tecla, se comprueba si el carácter de esa tecla se encuentra dentro de los caracteres permitidos para ese elemento <input>.

Si el carácter se encuentra dentro de los caracteres permitidos, se devuelve true y por tanto el comportamiento de onkeypress es el habitual y la tecla se escribe. Si el carácter no se encuentra dentro de los caracteres permitidos, se devuelve false y por tanto se impide el comportamiento normal deonkeypress y la tecla no llega a escribirse en el input.

Además, el script anterior siempre permite la pulsación de algunas teclas *especiales*. En concreto, las teclas BackSpace y Supr para borrar caracteres y las teclas Flecha Izquierda y Flecha Derecha para moverse en el cuadro de texto siempre se pueden pulsar independientemente del tipo de caracteres permitidos.