IES MARQUES DE COMARES

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Junta de Andalucía
Consejería de Educación y Deporte

Unión Europea-
NextGenerationEU.

UNIDAD 7

Formularios

IES MARQUES DE COMARES

ÍNDICE:

IES MARQUES DE COMARES

1.- Introducción

La validación de datos en aplicaciones Web es una de las tareas fundamentales para la programación en Javascript. Tanto es así que una de las principales razones por las que se inventó el lenguaje de programación JavaScript fue la de poder validar los datos de los formularios en el navegador del cliente, antes de realizar una comunicación con el servidor de la aplicación, porque se ahorraba mucho tiempo al no tener que recargar la página cuando el formulario contenía errores.

Hoy día, el principal uso de JavaScript es el de las comunicaciones asíncronas con los servidores, fundamentalmente para el acceso a bases de datos remotas y el de la manipulación dinámica de las aplicaciones

IES MARQUES DE COMARES

2.- Comunicación Síncrona y Asícrona

Comunicación Síncrona: Es aquella que se produce cuando el usuario necesita comunicarse con el servidor donde está alojada la aplicación Web, de forma que el navegador envía una petición al servidor, permaneciendo bloqueado hasta que el servidor procesa la petición y devuelve el código HTML como respuesta. Ej: PHP

Comunicación Asíncrona: Permite recargar en segundo plano una parte de la página web, dejando desbloqueado el resto. El cliente que envía una petición no permanece bloqueado esperando la respuesta del servidor, lo que supone un aumento de rapidez y de interactividad similar a una aplicación de escritorio. Ej. Ajax

3.- Formularios en javascript

Mediante JavaScript disponemos de gran facilidad para el manejo de formularios, ya que posee funciones y eventos que facilitan la programación de aplicaciones que manejan formularios.

3.- Acceso a formularios en Javascript

Los formularios son un elemento importante para dotar de interactividad a una aplicación Web, por lo que es muy importante el uso de uno o varios formularios en una página Web.

Ahora bien, para realmente poder dotar de interactividad a la página, debemos de poder acceder con Javascript tanto a cada uno de los formularios presentes en la página, como a cada uno de los elementos contenidos en cada uno de esos formularios, para así poder manipularlos de forma individual.

3.1 Acceso a través del objeto Forms

Cuando en un navegador se carga una página Web, éste crea de forma automática un array llamado forms mediante el que podemos referirnos a cualquiera de los formularios contenidos en la página.

Podemos acceder al contenido del array forms a través del objeto document, ya que document.forms es el array que contiene todos los formularios de la página, que pueden ser accedidos por su índice.

Por ejemplo, para acceder al primer formulario de una página basta con referirnos a él como

document.forms[0];

IES MARQUES DE COMARES

Acceso a través del objeto Forms

Del mismo modo, al cargar una página Web el navegador también de forma automática crea un array llamado elements, mediante el que podemos referirnos por su índice a cada uno de los elementos de cada uno de los formularios de nuestra página (cuadros de texto, botones, listas desplegables, etc.)

Por ejemplo, para acceder al primer y último elemento del primer formulario, así como al primer y último elemento del último formulario de la página basta con referirnos a ellos como:

```
document.forms[0].elements[0];
document.forms[0].elements[document.forms[0].elements.length-1];
document.forms[document.forms.length-1].elements[0];
document.forms[document.forms.length-1].elements[document.forms[document.forms.length-1].elements.length-1];
```

IES MARQUES DE COMARES

Acceso a través del objeto Forms

Sin embargo, acceder a los formularios mediante el array forms presenta un gran inconveniente:

- Si en un momento dado, cambiamos el diseño de la página, puede que también cambiemos el orden de los formularios originales o incluso que añadamos e eliminamos alguno de esos formularios, por lo que desde ese momento el orden ya no es el que estaba establecido en un primer momento y es posible que tengamos código javascript programado para el orden de los formularios de la página Web, que habría que retocar con menor o mayor dificultad.

Por tanto, este método de acceso, aunque está disponible es preferible no usarlo si esperamos modificaciones en el diseño.

3.1 Acceso a través de la propiedad “name” o “id”

El método más eficaz para acceder a los formularios de una página Web, es a través de su nombre con la propiedad “name” o a través de la propiedad ID con el método getElementById del DOM

Así por ejemplo, para acceder a un formulario de nombre “form1” e id “formulario1” podríamos usar cualquiera de las dos formas siguientes a través del objeto document:

```
var primerFormulario= document.form1;

var primerFormulario = document.getElementById("formulario1");
```

Acceso a través de la propiedad “name” o “id”

Del mismo modo, podemos acceder a cada uno de los elementos de un formulario a través de su “name” o su “id”. Por ejemplo para acceder al elemento1 con name “elem1” e id “elemento1” del formulario de name “form1” e id “formulario1”, podríamos usar cualquiera de las dos formas siguientes:

```
var primerFormulario= document.form1;
var primerElemento = document.form1.elem1

var primerFormulario = document.getElementById("formulario1");
var primerElemento = document.getElementById("elem1");
```

4.- Propiedades de los elementos de un formulario

type

Indica el tipo de elemento que se trata:

- Para los elementos de tipo <input> (text, button, checkbox, etc.) coincide con el valor de su atributo type.
- Para las listas desplegables , el valor de type es select-one
- Para las listas de selección múltiple, su valor es select-multiple.
- Para los elementos de tipo <textarea>, el valor de type es textarea.

form

Sirve para acceder al formulario al que pertenece un elemento, mediante la sintaxis: document.getElementById("id_del_elemento").form

Propiedades de los elementos de un formulario

name

obtiene el valor del atributo name de XHTML. Solamente se puede leer su valor, por lo que no se puede modificar.

value

permite leer y modificar el valor del atributo value de XHTML.

Por ejemplo, para los campos de texto (<input type="text"> y <textarea>) obtiene el texto que ha escrito el usuario, mientras que para los botones obtiene el texto que se muestra en el botón

Recordemos los eventos más usados con formularios (vistos en el tema anterior):

onclick: evento que se produce cuando se pincha con el ratón sobre un elemento. Normalmente se usa para los de tipo <input type="button">, <input type="submit"> y <input type="image">

onchange: evento que se produce cuando el usuario cambia el valor de un elemento de texto o cuando el usuario selecciona una opción en una lista desplegable. Sin embargo, el evento sólo se produce si el elemento pierde el focus, es decir, pasamos a otro elemento.

onfocus: evento que se produce cuando el usuario selecciona un elemento del formulario.

onblur: evento complementario de onfocus, ya que se produce cuando el usuario ha *deseleccionado* un elemento por haber seleccionado otro elemento del formulario y éste pierde el focus

5.- Acceso a los elementos de un formulario

- **Cuadro de texto y textarea**

Su contenido lo obtenemos mediante la propiedad value.

```
<input type="texto1" id="texto" >
var texto = document.getElementById("texto1").value;

<textarea id=“texto2"></textarea>
var texto = document.getElementById(“texto2").value
```

Accseso a los elementos de un formulario

Radiobutton

Interesa conocer cuál de todos los *radiobuttons* se ha seleccionado mediante la propiedad checked.

```
<input type="radio" value=“hombre” name=“sexo" id=“hombre”/> Hombre
<input type="radio" value=“mujer" name=“sexo" id=“mujer"/> Mujer

var sexo = document.getElementsByName(“sexo");
for(var i=0; i<sexo.length; i++)
 {
 if (sexo[i].checked) document.write(“Sexo:” +sexo[i].value);
}
```

Accseso a los elementos de un formulario

Checkbox

Interesa comprobar cada una de las opciones seleccionadas

```
<input type="checkbox" value=“futbol" name=“futbol" id=“fubol"/> Fútbol
 <input type="checkbox" value=“tenis" name=“tenis" id=“tenis"/> Tenis

var elemento = document.getElementById(“futbol");

 if (elemento.checked) document.write(elemento[i].value);
 elemento = document.getElementById(“tenis");
 if (elemento.checked) document.write(elemento[i].value);
```

Accseso a los elementos de un formulario

Select

Interesa obtener el valor de la opción seleccionada por el usuario

```
<select id="aficiones" name="aficiones">
<option value="fútbol">Fútbol</option>
<option value="tenis">Tenis</option>
<option value="baloncesto">Baloncesto</option>
</select>
```

- options, array creado automáticamente por el navegador para cada lista desplegable. Por ejemplo, la primera opción de la lista se puede obtener mediante document.getElementById("aficiones").options[0].
- selectedIndex, cuando el usuario selecciona una opción, el navegador guarda el índice de la opción seleccionada. Para obtener el texto y el valor del elemento inmediatamente seleccionado:

```
var lista = document.getElementById("aficiones");
var valorSeleccionado = lista.options[lista.selectedIndex].value;   //valor
var valorSeleccionado = lista.options[lista.selectedIndex].text;    // texto
```

6.- Trabajar con el focus

En Javascript, decimos que un elemento posee el focus, cuando está seleccionado y se puede escribir directamente en él.
Por ejemplo, si un cuadro de texto de un formulario tiene el focus, el usuario puede escribir en dicho cuadro de texto, o cuando una lista desplegable tiene el focus, podremos seleccionar una opción de la misma desplazándose con las teclas del cursor
En una página Web, al pulsar la tecla TABULADOR vamos avanzando en el focus de cada uno de los elementos de la página según el orden que hayamos establecido de los mismos, de forma que el elemento seleccionado suele mostrar algún borde que indique que posee el focus.

Trabajar con el focus

Normalmente, en una página Web con formulario, se suele asignar el focus al cargar la página al primer elemento del formulario.

Para asignar el focus a un elemento de XHTML, se utiliza la función focus().

Ejemplo: Asignar el focus a un elemento del formulario con id="nombre".

```
document.getElementById("nombre").focus();
```

Sin embargo, es muy posible que con el tiempo no sepamos el id del primer elemento (hemos cambiado, añadido o eliminado elementos al formulario). Para ello, podemos asignar asignar automáticamente el focus del programa al primer elemento del primer formulario de nuestra página (siempre que no sea del tipo hidden, sin tener que saber nada de sus names o ids.

```
if(document.forms.length > 0) {
        if(     document.forms[0].elements.length > 0  &&
                document.forms[0].elements[0].type != "hidden) {
        document.forms[0].elements[0].focus();
        }
}
```

Trabajar con el focus

Uno de las situaciones más habituales en los formularios es dar varias veces al botón “Enviar” porque creamos que nuestro formulario aún no ha sido enviado.
Para evitar problemas, es útil deshabilitar el botón de envío después de la primera vez que hayamos pulsado el botón “Enviar”.

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

Nota: no funciona bien con los botones tipo submit

Trabajar con el focus

JavaScript permite comprobar si hemos alcanzado el número máximo de caracteres en un textarea (cosa que no podemos hacer en un formulario).
Para ello, podemos crear una función similar a esta y hacer uso de la misma:

```
function maximocaracteres(maximoCaracteres) {
var elemento = document.getElementById("texto1");
if(elemento.value.length >= maximoCaracteres ) {
        return false;
        }
else {
        return true;
        }
}
<textarea id="texto1" onkeypress="return maximocaracteres(50);"></textarea>
```

Trabajar con el focus

```
function permitir(elEvento, permitidos) {
var numeros="0123456789";
var letras = " abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ";
var caracteres_validos= numeros + letras;
// 8 = BackSpace, 46 = Supr, 37 = flecha izquierda, 39 = flecha derecha
var especiales = [8, 37, 39, 46];
// Seleccionar los caracteres a partir del parámetro de la función
 switch(permitidos) {
        case 'num':
                permitidos = numeros;
                break;
        case 'letra':
                permitidos = caracteres;
                break;
        case 'numletra':
                permitidos =caracteres_validos;
                break;
        }
```

Trabajar con el focus

```
// 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
var tecla_especial = false;
for(var i in especiales) {
        if(codigoCaracter == especiales[i]) {
        tecla_especial = true;
        break;
        }
}
// Comprobar si la tecla pulsada se encuentra en los caracteres permitidos
return permitidos.indexOf(caracter) != -1 || tecla_especial;
}
```

Trabajar con el focus

```
// Sólo números
<input type="text" id="texto" onkeypress="return permitir(event, 'num')" />

// Sólo letras
<input type="text" id="texto" onkeypress="return permitir(event, ‘letra')" />

// Sólo letras o números
<input type="text" id="texto" onkeypress="return permitir(event, ‘numletra')" />
```

7.- Validación de formularios

Validar los datos introducidos por los usuarios en los elementos de un formulario es fundamental para detectar si se ha cometido algún error al rellenar el formulario, y se le pueda notificar sin esperar la respuesta del servidor.

Normalmente, la validación de un formulario consiste en llamar a una función de validación cuando el usuario pulsa sobre el botón enivar, comprobando si los valores de los elementos del formulario cumplen las reglas de validación de dichos elementos.

Aunque existen tantas posibles comprobaciones como elementos de formulario diferentes, algunas comprobaciones son muy habituales: que se rellene un campo obligatorio, que se seleccione el valor de una lista desplegable, que la dirección de email indicada sea correcta, que la fecha introducida sea lógica, que se haya introducido un número donde así se requiere, etc.

Validación de formularios

```
<form action=""  id="" name="" onsubmit="return validacion()">
// si evento onsubmit devuelve el valor true, el formulario se envía,pero si devuelve false, no se envía
 ...
</form>
// Esquema general funcion validación
function validacion() {
if (not condicion primer campo) {
            alert('[ERROR]...');
            return false;
            }
 if (not condicion segundo campo) {
            alert('[ERROR]...');
            return false;
            } …
if (not condicion último campo) {
                        alert('[ERROR]...');
                        return false;
                        }
return true; }
```

7.1.- Validar campo de texto obligatorio

Habrá que validar si se ha introducido un valor en un cuadro de texto o textarea que era obligatorio. Para ello se comprueba que el valor introducido sea válido, que haya introducido al menos 1 carácter y que no se hayan introducido sólo espacios en blanco

Nota: Repasar expresiones regulares
https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_Expressions

```
valor = document.getElementById("campo").value;
if( valor == null || valor.length == 0 ||  /^\s+$/.test(valor) ) {
        return false;
        }
//null indica
```

7.2.- Validar campo de texto numérico

Habrá que validar si se ha introducido un valor numérico en un cuadro de texto.

```
valor = document.getElementById("campo").value;
if( isNaN(valor) ) {
        return false;
        }
```

7.3.- Validar selección de lista

Habrá que validar se ha seleccionado un elemento de una lista desplegable.

```
indice = document.getElementById(“lista").selectedIndex;
 if( indice == null || indice == 0 ) {
            return false;
            }
```

```
<select id=“lista" name=“lista">
<option value=“futbol">Fútbol</option>
<option value=“tenis">Tenis</option>
</select>
```

7.4.- Validar un checkbox

Habrá que validar si un elemento (o varios) de tipo *checkbox* se ha seleccionado.

```
elemento = document.getElementById("campo");
if( !elemento.checked ) {
        return false;
        }
// para comprobar varios
formulario = document.getElementById("formulario");
for(var i=0; i<formulario.elements.length; i++) {
        var elemento = formulario.elements[i];
        if(elemento.type == "checkbox") {
                if(!elemento.checked) {
                        return false;
                }
        }
}
```

7.5.- Validar un radiobutton

Habrá que validar se se ha seleccionado algún *radiobutton.*

```
opciones = document.getElementsByName(“radiobutton");
var seleccionado = false;
for(var i=0; i<opciones.length; i++) {
        if(opciones[i].checked) {
                seleccionado = true;
                break;
        }
}
if(!seleccionado) {
        return false;
        }
```

7.6.- Validar dirección email

Habrá que validar si la dirección de correo tiene un formato válido.

Nota: Repasar expresiones regulares
https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_Expressions

```
valor = document.getElementById("campo").value;
if( !(/\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)/.test(valor)) ) {
        return false;
        }
```

7.7.- Validar una fecha

Habrá que validar si la fecha tiene un formato válido.

```
var ano = document.getElementById("ano").value;
var mes = document.getElementById("mes").value;
var dia = document.getElementById("dia").value;
valor = new Date(ano, mes, dia);
if( !isNaN(valor) ) {
        return false;
        }
```