Manejo de formularios con JavaScript

La mayor parte de interactividad entre una página web y el usuario tiene lugar a través de un formulario. Es ahí donde nos vamos a encontrar con los campos de texto, botones, checkboxes, listas, etc. en los que el usuario introducirá los datos, que luego se enviarán al servidor.

En este apartado se verá cómo identificar un formulario y sus objetos, cómo modificarlos, cómo examinar las entradas de usuario, enviar un formulario, validar datos,..etc.

Los formularios y sus controles, son objetos **DOM** que tienen propiedades únicas, que otros objetos no poseen. Por ejemplo, un formulario tiene una propiedad **action**, que le indica al navegador dónde tiene que enviar las entradas del usuario, cuando se envía el formulario. Un control **select** posee una propiedad llamada **selectedIndex**, que nos indica la opción de ese campo que ha sido seleccionada por el usuario.

JavaScript añade a los formularios dos características muy interesantes:

- JavaScript permite examinar y validar las entradas de usuario directamente, en el lado del cliente.
- JavaScript permite dar mensajes instantáneos, con información de la entrada del usuario.

El objeto de JavasScript **form**, es una propiedad del objeto **document**. Se corresponderá con la etiqueta **<FORM>** del HTML. Un formulario podrá ser enviado llamando al método **submit** de JavaScript, o bien haciendo **click** en el botón **submit** del formulario.

En la siguiente web se puede ver una lista de los objetos utilizados al trabajar con formularios.

Sumario

- 1 Formas de selección del objeto form
- 2 El formulario como objeto y contenedor
- 3 Acceso a propiedades y métodos del formulario
    - 3.1 Propiedad form.elements[ ]
- 4 Objetos relacionados con formularios
    - 4.1 Objeto input de tipo texto
    - 4.2 Objeto *input* de tipo *checkbox*
    - 4.3 Objeto input de tipo radio
    - 4.4 Objeto select
- 5 Empleando el identificador **this**
- 6 Envío y validación de formuarios
    - 6.1 Ejemplo sencillo de validación de un formulario

Formas de selección del objeto form

Dentro de un documento tendremos varias formas de selección de un formulario. Si partimos del siguiente ejemplo:

```
<div  id="menulateral">
    <form  id="contactar" name="contactar" action="...">...</form>
</div>
...
```

Tendremos los siguientes métodos de selección del objeto **form** en el documento:

> Método 1

A través del método getElementById() del DOM, nos permite acceder a un objeto a través de su atributo ID.

```
var formulario=document.getElementById("contactar");
```

> Método 2

A través del método getElementsByTagName() del DOM, el cual nos permite acceder a un objeto a través de la etiqueta HTML que queramos. Por ejemplo para acceder a los objetos con etiqueta **form** haremos:

```
var formularios = document.getElementsByTagName('form');
var primerFormulario = formularios[0];// primer formulario del documento
```

Otra posibilidad interesante que te permite el método anterior, es la de **buscar objetos con un padre determinado,** por ejemplo:

```
var menu = document.getElementById("menulateral");
// formularios contenidos en 'menulateral'
var formularios = menu.getElementsByTagName("form");
// primer formulario en el menú lateral
var primerFormulario= formularios[0];
```

> Método 3

Otro método puede ser a través de la colección forms[ ] del objeto **document**. Esta colección es un array, que contiene la referencia a todos los formularios que tenemos en nuestro documento.

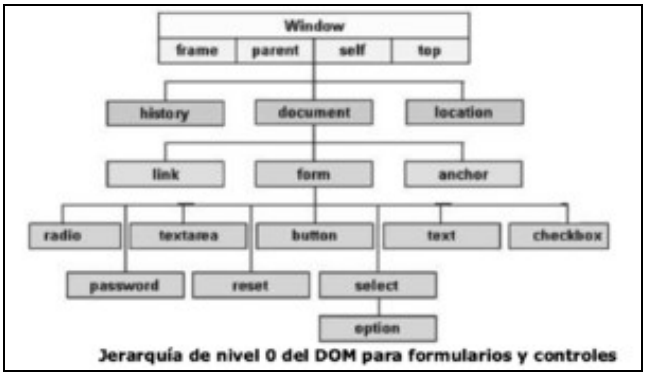
```
Por ejemplo:
// la referencia a todos los formularios del documento
var formularios = document.forms;
// primer formulario del documento
var miformulario = formularios[0];

o bien:
// primer formulario del documento
var miformulario = document.forms[0];

o bien:
// referenciamos al formulario con name "contactar"
var miformulario = formularios["contactar"];
```

El formulario como objeto y contenedor

Debido a que el DOM ha ido evolucionando con las nuevas versiones de JavaScript, nos encontramos con que el objeto **form** está dentro de dos árboles al mismo tiempo. En las nuevas definiciones del DOM, se especifica que **form** es el padre de todos sus nodos hijos, incluidos objetos y textos, mientras que en las versiones antiguas **form** sólo era padre de sus objetos (**input**, **select**, **button** y elementos **textarea**).



Jerarquía de nivel 0 del DOM para formularios y controles

form DOM

Veamos un ejemplo de cómo sería el árbol DOM para un formulario típico:

```
<form action="buscar.php" name="elFormulario" id="miFormulario" method="post">
<p>
<label for="busqueda">Buscar por:</label>
<input id="busqueda" name="busqueda" type="text" value="" />
<input id="submit" type="submit" value="Buscar" />
</p>
</form>

<script type="text/javascript">
function printDOM(node, prefix) {
console.log(`${prefix} ${node.nodeName}  :  ${node.nodeType}  ->  ${node.nodeValue}`);
for (let i = 0; i < node.childNodes.length; i++) {
printDOM(node.childNodes[i], `${prefix} \t`);
}
}
            printDOM(document.getElementById("miFormulario"), "");

    //Salida:
    //     FORM  :  1  ->  null
//  #text  :  3  ->
//  P  :  1  ->  null
//       #text  :  3  ->
//       LABEL  :  1  ->  null
//               #text  :  3  ->  Buscar por:
//       #text  :  3  ->
//       INPUT  :  1  ->  null
//       #text  :  3  ->
//       INPUT  :  1  ->  null
//       #text  :  3  ->
```

```
//  #text  :  3  ->
</script>
</body>
```

Acceso a propiedades y métodos del formulario

Los formularios pueden ser creados a través de las etiquetas HTML, o utilizando JavaScript y métodos del DOM. En cualquier caso se pueden asignar atributos como **name**, **action**, **target** y **enctype**. Cada uno de estos atributos es una propiedad del objeto **form**, a las que podemos acceder utilizando su nombre en minúsculas, por ejemplo:

```
var paginaDestino = objetoFormulario.action;
```

Para modificar una de estas propiedades lo haremos mediante asignaciones, por ejemplo:

```
objetoFormulario.action = "http://www.xunta.es/recepcion.php";
```

Estas dos instrucciones las podemos recomponer usando referencias a objetos:

```
var paginaDestino = document.getElementByid("id").action;
document.forms[0].action = "http://www.xunta.es/recepcion.php";
```

Propiedades del objeto form

Propiedad	Descripción	W3C
acceptCharset	Ajusta o devuelve el valor del atributo accept-charset en un formulario.	Si
action	Ajusta o devuelve el valor del atributo action en un formulario.	Si
enctype	Ajusta o devuelve el valor del atributo enctype en un formulario.	Si
length	Devuelve el número de elementos en un formulario.	Si
method	Ajusta o devuelve el valor del atributo method en un formulario.	Si
name	Ajusta o devuelve el valor del atributo name en un formulario.	Si
target	Ajusta o devuelve el valor del atributo target en un formulario.	Si

Métodos del objeto form

Método	Descripción	W3C
reset()	Resetea un formulario.	Si
submit()	Envía un formulario.	Si

Propiedad form.elements[ ]

La propiedad **elements[ ]** de un formulario es una colección, que contiene todos los objetos **input** dentro de un formulario. Esta propiedad es otro *array*, con todos los campos **input** en el orden en el cual aparecen en el código fuente del documento.

Generalmente, es mucho más eficaz y rápido referenciar a un elemento individual usando su ID, pero a veces, los *scripts* necesitan recorrer cada elemento del formulario, para comprobar que se han introducido sus valores correctamente.

Por ejemplo, empleando la propiedad **elements[]**, podemos hacer un bucle que recorra un formulario y muestre el 'id' de cada elemento acompañado del tipo de elemento de que trata:

```
// Guardamos la referencia del formulario en una variable.
const miFormulario = document.getElementById("formulario");

// Si no existe ese formulario devuelve ''false''.
if (!miFormulario) {
    console.log('No existe ningún formulario');
}
else {
    for (let i=0; i< miFormulario.elements.length; i++) {
        let tipoElemento = miFormulario.elements[i].type;
        let idElemento = miFormulario.elements[i].id;
        console.log(`${idElemento} : ${tipoElemento}`);
    }
}
```

Objetos relacionados con formularios

Para poder trabajar con los objetos de un formulario, lo primero que necesitas saber es, cómo referenciar a ese objeto. Eso puedes hacerlo directamente a través de su **ID**, o bien con su nombre de etiqueta, empleando para ello los métodos del DOM nivel 2. O también se puede hacer usando la sintaxis del DOM nivel 0, y construyendo la jerarquía que comienza por **document**, luego el formulario y finalmente el control.

Lo mejor es identificar cada uno de los objetos con un atributo **id** que sea único, y que no se repita en el documento, así para acceder a cualquier objeto dentro de nuestro documento o formulario lo haremos con:

```
document.getElementById("id-del-control")
// O, de otro modo
document.nombreFormulario.name-del-control
```

Por ejemplo, si consideramos un formulario sencillo como el siguiente:

```
<form  id="formularioBusqueda"  action="cgi-bin/buscar.pl">
  <p>
    <input  type="text"  id="entrada"  name="cEntrada">
    <input  type="submit"  id="enviar"
name="enviar"  value="Buscar...">
  </p>
</form>

// Las siguientes referencias al campo de texto entrada, serán todas válidas:

document.getElementById("entrada")
document.formularioBusqueda.cEntrada
document.formularioBusqueda.elements[0]
document.forms["formularioBusqueda"].elements["cEntrada"]
document.forms["formularioBusqueda"].cEntrada
```

Aunque muchos de los controles de un formulario tienen propiedades en común, algunas propiedades son únicas a un control en particular. Por ejemplo, en un objeto **select** tienes propiedades que te permiten conocer la opción que está actualmente seleccionada. Al igual que los *checkboxes* o los botones de tipo radio, que también disponen de propiedades para saber cuál es la opción que está actualmente seleccionada.

Objeto input de tipo texto

Cada uno de los 4 elementos de tipo texto de los formularios: *text*, *password*, *hidden* y elementos *textarea*, son un elemento dentro de la jerarquía de objetos. Todos los elementos, excepto los tipos *hidden*, se mostrarán en la página, permitiendo a los usuarios introducir texto y seleccionar opciones.

Para poder usar estos objetos dentro de nuestros scripts de JavaScript, simplemente será suficiente con asignar un atributo **id**, a cada uno de los elementos. Te recomiendo que asignes a cada objeto de tu formulario un atributo **id** único y que coincida con el *name* de ese objeto.

Cuando se envían los datos de un formulario a un programa en el lado del servidor, lo que en realidad se envía son los atributos *name*, junto con los valores (contenido del atributo *value*) de cada elemento. Sin lugar a dudas, la propiedad más utilizada en un elemento de tipo texto es por lo tanto *value*. Un *script* podrá recuperar y ajustar el contenido de la propiedad *value* en cualquier momento. Por cierto, el contenido de un *value* es siempre una cadena de texto, y quizás puedas necesitar realizar conversiones numéricas si quieres realizar operaciones matemáticas con esos textos.

En este tipo de objetos, los gestores de eventos (que verás más adelante) se podrán disparar de múltiples formas: por ejemplo, cuando pongamos el foco en un campo (situar el cursor dentro de ese campo) o modifiquemos el texto (al introducir el texto y salir del campo).

Propiedades de los input de tipo texto

Propiedad	Descripción	W3C
defaultValue	Ajusta o devuelve el valor por defecto de un campo de texto.	Si
form	Devuelve la referencia al formulario que contiene ese campo de texto.	Si
maxLength	Devuelve o ajusta la longitud máxima de caracteres permitidos en el campo de tipo texto	Si
name	Ajusta o devuelve el valor del atributo name de un campo de texto.	Si
readOnly	Ajusta o devuelve si un campo es de sólo lectura, o no.	Si
size	Ajusta o devuelve el ancho de un campo de texto (en caracteres).	Si

Propiedad	Descripción	W3C
type	Devuelve el tipo de un campo de texto.	Si
value	Ajusta o devuelve el contenido del atributo value de un campo de texto.	Si

Métodos de los objetos input de tipo texto

Método	Descripción	W3C
select()	Selecciona el contenido de un campo de texto.	Si

Además de los métodos anteriores, los campos de tipo texto también soportan todas las propiedades estándar, métodos y eventos.

Objeto *input* de tipo *checkbox*

Un **checkbox** es también un objeto muy utilizado en los formularios, pero algunas de sus propiedades puede que no sean muy intuitivas. En los botones de un formulario la propiedad **value** nos mostrará el texto del botón, pero en un **checkbox** la propiedad **value** es un texto que está asociado al objeto. Este texto no se mostrará en la página, y su finalidad es la de asociar un valor con la opción actualmente seleccionada. Dicho valor será el que se enviará, cuando enviemos el formulario. Veamos un ejemplo:

```
<label  for="cantidad">
  Si desea recibir 20 Kg marque esta opción:
</label>
<input  type="checkbox"  id="cantidad"  name="cantidad"  value="20 Kg">
```

Si chequeamos este **checkbox** y enviamos el formulario, el navegador enviará el par **name**/**value** "cantidad y "20 Kg". Si el **checkbox** no está marcado, entonces este campo no será enviado en el formulario. El texto del **label** se mostrará en la pantalla pero las etiquetas **label** no se envían al servidor. Para saber si un campo de tipo **checkbox** está o no marcado, disponemos de la propiedad **checked**. Esta propiedad contiene un valor booleano: **true** si el campo está marcado o **false** si no está marcado. Con esta propiedad es realmente sencillo comprobar o ajustar la marca en un campo de tipo **checkbox**.

Veamos el siguiente ejemplo:

```
<html>
  <head>
    <title>Ejemplo formularios</title>
  </head>

  <body>
    <form action="" method="get">
      <input type="checkbox" id="verano" name="verano" value="Si"/>¿Te gusta el verano?
      <input type="submit" />
    </form>
    <button onclick="marcar()">Marcar Checkbox</button>
    <button onclick="desmarcar()">Desmarcar Checkbox</button>

  <script type="text/javascript">
    function marcar() {
      document.getElementById("verano").checked=true;
    }

    function desmarcar() {
      document.getElementById("verano").checked=false;
    }
  </script>


  </body>
</html>
```

Propiedades de los objetos input de tipo checkbox

Propiedad	Descripción	W3C
checked	Ajusta o devuelve el estado checked de un checkbox.	Si
defaultChecked	Devuelve el valor por defecto del atributo checked.	Si
form	Devuelve la referencia al formulario que contiene ese campo checkbox.	Si

Propiedad	Descripción	W3C
name	Ajusta o devuelve el valor del atributo name de un checkbox.	Si
type	Nos indica que tipo de elemento de formulario es un checkbox.	Si
value	Ajusta o devuelve el valor del atributo value de un checkbox.	Si

Objeto input de tipo radio

El ajustar un grupo de objetos de tipo radio desde JavaScript requiere un poco más de trabajo. Para dejar que el navegador gestione un grupo de objetos de tipo radio, deberemos asignar el mismo atributo **name** a cada uno de los botones del grupo. Podemos tener múltiples grupos de botones de tipo radio en un formulario, pero cada miembro de cada grupo tendrá que tener el mismo atributo **name** que el resto de compañeros del grupo.

Cuando le asignamos el mismo **name** a varios elementos en un formulario, el navegador lo que hace es crear un *array* con la lista de esos objetos que tienen el mismo **name**. El contenido del atributo **name** será el nombre del *array*. Algunas propiedades, se las podremos aplicar al grupo como un todo; otras en cambio, tendremos que aplicárselas a cada elemento del grupo y lo haremos a través del índice del *array* del grupo. Por ejemplo, podemos ver cuantos botones hay en un grupo radio, consultando la propiedad *length* de ese grupo:

```
objetoFormulario.nombregrupo.length
```

Y si queremos acceder a la propiedad **checked** de un botón en particular, lo haremos accediendo a la posición del *array* y a la propiedad **checked**:

```
// Accedemos a la propiedad checked del primer botón del grupo
objetoFormulario.nombregrupo[0].checked
```

Veamos un ejemplo:

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>DWCC - Trabajando con objetos input de tipo radio</title>
</head>
<body>
  <h1>Trabajando con objetos input de tipo radio</h1>
  <form  name="formulario" action="stooges.php">
    <fieldset><legend>Selecciona tu actor favorito:</legend>
      <input  type="radio"  name="actores"  id="actor-1" value="Walter Bruce Willis - 19 de Marzo de 1955"  checked>
        <label  for="actor-1">Willis</label>
      <input  type="radio"  name="actores"  id="actor-2" value="James Eugene Jim Carrey - 17 de Enero de 1962">
        <label  for="actor-2">Carrey</label>
      <input  type="radio"  name="actores"  id="actor-3" value="Luis Tosar - 13 de Octubre de 1971">
        <label  for="actor-3">Tosar</label>
      <input  type="button"  id="consultar"  name="consultar" value="Consultar Más Datos" onclick="mostrarDatos()">
    </fieldset>
  </form>

  <script type="text/javascript">
    function  mostrarDatos() {
      for  (let i=0; i<document.formulario.actores.length; i++) {
        if  (document.formulario.actores[i].checked) {
  alert(document.formulario.actores[i].value);
}
      }
    }
  </script>

</body>
</html>
```

Objeto select

Uno de los controles más complejos que te puedes encontrar en formularios, es el objeto **select**. Un objeto **select** está compuesto realmente de un *array* de objetos option. El objeto **select** se suele mostrar como una lista desplegable en la que seleccionas una de las opciones, aunque también

tienes la opción de selecciones múltiples, según definas el objeto en tu documento. Vamos a ver cómo gestionar una lista que permita solamente selecciones sencillas.

Algunas propiedades pertenecen al objeto **select** al completo, mientras que otras, por ejemplo, sólo se pueden aplicar a las opciones individuales dentro de ese objeto. Si lo que quieres hacer es detectar la opción seleccionada por el usuario, y quieres usar JavaScript, tendrás que utilizar propiedades tanto de **select**, como de **option**. La propiedad más importante del objeto **select** es la propiedad **selectedIndex**, a la que puedes acceder de las siguientes formas:

```
objetoFormulario.nombreCampoSelect.selectedIndex;
document.getElementById("objetoSelect").selectedIndex;
```

El valor devuelto por esta propiedad, es el índice de la opción actualmente seleccionada y, por supuesto, recordarte que los índices comienzan en la posición 0. Siempre que queramos acceder a la opción actualmente seleccionada, recuerda que tendrás que usar esta propiedad.

Las opciones tienen dos propiedades accesibles que son **text** y **value**, y que te permitirán acceder al texto visible en la selección y a su valor interno para esa opción. Podemos ver un ejemplo:

```
<!DOCTYPE  html>
<html>
  <head>
    <meta  http-equiv="content-type"  content="text/html;charset=utf-8">
    <title>DWCC - Trabajando con un objeto Select</title>
  </head>

  <body>
    <h1>Trabajando con un objeto Select</h1>
    <form id="formulario" action="pagina.php">
      <p>
        <label for="provincias">Seleccione provincia: </label>
        <select name="provincias" id="provincias">
          <option value="C">La Coruña</option>
          <option value="LU">Lugo</option>
          <option value="OU">Ourense</option>
          <option value="PO">Pontevedra</option>
        </select>
      </p>
      <p>Selecciona una opción y pulsa el botón.</p>
      <input type="button" name="boton" value="Consultar información de la opción" onclick="consultar()"/>
    </form>

  <script  type="text/javascript">
    function consultar() {
      let objProvincias=document.getElementById("provincias");
      let texto=objProvincias.options[objProvincias.selectedIndex].text;
      let valor=objProvincias.options[objProvincias.selectedIndex].value;
      alert("Datos de la opción seleccionada:\n\nTexto: " + texto + "\nValor: " + valor);
  }
  </script>

  </body>

</html>
```

Empleando el identificador this

En los ejemplos que hemos visto anteriormente cuando un gestor de eventos (**onclick**, **onblur**,...) llama a una función, esa función se encarga de buscar el objeto sobre el cuál va a trabajar. En JavaScript disponemos de un método que nos permite llamar a una función, pasándole directamente la referencia del objeto, sin tener que usar variables globales o referenciar al objeto al comienzo de cada función.

Para conseguir hacerlo necesitamos usar la palabra reservada **this**, la cuál hace referencia siempre al objeto que contiene el código de JavaScript en donde usamos dicha palabra reservada. Por ejemplo, si programamos una función para un botón, que al hacer **click** haga algo, si dentro de esa función usamos la palabra **this**, entonces estaremos haciendo referencia al objeto en el cuál hemos hecho **click** que, en este caso, será el botón. El uso de **this** nos permite evitar usar variables globales, y el programar *scripts* más genéricos.

Por ejemplo:

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<title>Uso de la palabra reservada this</title>
</head>
<body>
<h1>Trabajando con this</h1>
<form id="formulario" action="pagina.php">
<p>
<label for="nombre">Nombre: </label>
<input
type="text"
name="nombre"
id="nombre"
value="Constantino"
/>
<label for="apellidos">Apellidos: </label>
<input
type="text"
name="apellidos"
id="apellidos"
value="Veiga Perez"
/>
<label for="edad">Edad: </label>
<input
type="password"
name="edad"
id="edad"
value="55"
/>
<label for="pais">Pais: </label>
                                España
<input
type="radio"
name="pais"
id="pais1"
value="ES"
/>
                                Francia
<input
type="radio"
name="pais"
id="pais2"
value="FR"
/>
</p>
                        Haga click en cada uno de los campos para ver más información.
</form>

        <script type="text/javascript">

            document.getElementById('nombre').addEventListener('click', identificar, false);
            document.getElementById('apellidos').addEventListener('click', identificar, false);
            document.getElementById('edad').addEventListener('click', identificar, false);
            document.getElementById('pais1').addEventListener('click', identificar, false);
            document.getElementById('pais2').addEventListener('click', identificar, false);

            function identificar() {
                let atrName = this.name;
                let atrId = this.id;
                let atrValue = this.value;
                let atrType = this.type;

                alert(
                    "Datos del campo pulsado:\n\nName: " +
                        atrName +
                        "\nID: " +
                        atrId +
                        "\nValue: " +
                        atrValue +
                        "\nType: " +
                        atrType
                );
```

```
        }
    </script>
```

En este ejemplo, cada vez que hagamos *click* en alguno de los objetos, llamaremos a la función **identificar()** en esa función se emplea el identificador **this**, que en este caso será la referencia al objeto en el cuál hemos hecho *click*. Así podremos imprimir todas las referencias al *name*, *id*, *value* y *type*.

Envío y validación de formuarios

La validación de un formulario es un proceso que consiste en chequear un formulario y comprobar que todos sus datos han sido introducidos correctamente. Por ejemplo, si tu formulario contiene un campo de texto en el que hay que escribir un e-mail, sería interesante comprobar si ese e-mail está escrito correctamente, antes de pasar al siguiente campo.

Hay dos métodos principales de validación de formularios: en el lado del servidor (usando *scripts* CGI, PHP, ASP, etc.) y en el lado del cliente (generalmente usando JavaScript).

La validación en el lado del servidor es más segura, pero a veces también es más complicada de programar, mientras que la validación en el lado del cliente es más fácil y más rápida de hacer (el navegador no tiene que conectarse al servidor para validar el formulario, por lo que el usuario recibirá información al instante sobre posibles errores o fallos encontrados).

La idea general que se persigue al validar un formulario, es que cuando se envíen los datos al servidor, éstos vayan correctamente validados, sin ningún campo con valores incorrectos.

A la hora de programar la validación, podremos hacerlo a medida que vamos metiendo datos en el formulario, por ejemplo campo a campo, o cuando se pulse el botón de envío del formulario.

JavaScript añade a tus formularios dos características muy interesantes:

- JavaScript te permite examinar y validar las entradas de usuario directamente, en el lado del cliente.
- JavaScript te permite dar mensajes instantáneos, con información de la entrada del usuario.

La validación de datos del usuario en la entrada, generalmente suele fallar en alguna de las 3 siguientes categorías:

- Existencia: comprueba cuando existe o no un valor.
- Numérica: que la información contenga solamente valores numéricos.
- Patrones: comprueba que los datos sigan un determinado patrón, como el formato de un e-mail, una fecha, un número de teléfono, un número de la seguridad social, etc.

JavaScript también se puede utilizar para modificar los elementos de un formulario, basándose en los datos introducidos por el usuario: tal como cubrir un campo de selección con una lista de nombres de ciudades, cuando una determinada provincia está seleccionada, etc.

Una parte muy importante que no debes olvidar al usar JavaScript con formularios, es la posibilidad de que el usuario desactive JavaScript en su navegador, por lo que JavaScript no debería ser una dependencia en la acción de envío de datos desde un formulario.

La validación de un formulario en el lado del cliente puede ahorrar algunas idas y vueltas a la hora de enviar los datos, pero, aún así, tendrás que realizar la validación de datos en el servidor, puesto que es allí realmente donde se van a almacenar esos datos y el origen de los mismos puede venir por cauces que no hemos programado.

Ejemplo sencillo de validación de un formulario

Como se comentó anteriormente el objetivo de validar un formulario es comprobar, que antes del envío del mismo, todos los campos poseen valores correctos, evitando así que el usuario tenga que volver de nuevo al formulario, si es que no se pudo validar correctamente en el servidor. Eso no evita que también tengamos que hacer validación en el servidor, ya que recuerda, que el usuario podrá desactivar JavaScript en su navegador, con lo que la validación que hemos programado en JavaScript no funcionaría.

Validación de Formularios

- **Código HTML:**

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Ejemplo de Validacion de un Formulario</title>

<style type="text/css">
label {
width: 150px;
float: left;
margin-bottom: 5px;
}
            input {
                float: left;
            }
input,
select {
width: 150px;
margin-bottom: 5px;
}
fieldset {
background: #ccff99;
width: 350px;
}
.error {
border: solid 2px #ff0000;
}
</style>
</head>
<body>
<fieldset>
<legend>Validación de un Formulario</legend>
<form name="formulario" id="formulario" action="http://www.laweb.es" method="get">
<label for="nombre">Nombre:</label>
             <input type="text" name="nombre" id="nombre" />
<label for="apellidos">Apellidos:</label>
             <input type="text" name="apellidos" id="apellidos" />
<label for="edad">Edad:</label>
             <input name="edad" type="text" id="edad" maxlength="3" />
<label for="matricula">Matrícula Coche:</label>
             <input name="matricula" type="text" id="matricula" value="XXXX AAA" />
<label for="provincia">Provincia:</label>
             <select name="provincia" id="provincia">
             <option value="0" selected="selected">Seleccione Provincia</option>
    <option value="H">Huesca</option>
    <option value="ZA">Zaragoza</option>
    <option value="T">Teruel</option>
                  </select>
               <label></label>
<input type="reset" name="limpiar" id="limpiar" value="Limpiar" />
<input type="submit" name="enviar" id="enviar" value="Enviar" />
</form>
        </fieldset>

        <script type="text/javascript" src="DWCC_ValidacionFormulario.js"></script>
```

```
</body>
</html>
```

- **Código JavaScript**

```
// En la variable que pongamos aquí gestionaremos el evento por defecto
function validar(e) {
// El evento 'e' está asociado al botón 'enviar' (type=submit)
// en este caso lo que hace es enviar el formulario

// Validamos cada una de las cajas de texto con llamadas a sus funciones correspondientes.
if (
validarCamposTexto(this) &&
validarProvincia() &&
confirm("¿Deseas enviar el formulario?")
)
return true;
else {
// Cancelamos el evento de envío por defecto asignado al botón de 'submit' enviar
e.preventDefault();
return false; //Salimos de la función devolviendo false
}
}
//-----------------------------------------------------------//

function validarCamposTexto(objeto) {
// A esta función se le pasa como parámetro el propio botón 'enviar' (this)

// La propiedad 'form' del botón 'enviar' contiene la referencia del formulario
let formulario = objeto.form;

// Recorremos todos los elementos del formulario
for (let i = 0; i < formulario.elements.length; i++) {
// Eliminamos la clase .error que estuviera asignada a algún campo
formulario.elements[i].classList.remove("error");

// Buscamos los que son de tipo texto
if (
formulario.elements[i].type == "text" &&
formulario.elements[i].value == ""
) {
alert(
"El campo: " +
formulario.elements[i].name +
" no puede estar en blanco"
);
formulario.elements[i].classList.add("error");
formulario.elements[i].focus();
return false;
}
// Validamos también el campo 'edad'
else if (formulario.elements[i].id == "edad") {
let laEdad = formulario.elements[i].value;
if (isNaN(laEdad) || laEdad < 0 || laEdad > 115) {
alert(
"El campo: " +
formulario.elements[i].name +
" posee valores incorrectos"
);
formulario.elements[i].classList.add("error");
formulario.elements[i].focus();
return false;
}
} else if (formulario.elements[i].id == "matricula") {
// 4 Números 1 espacio en blanco(opcional) y 3 letras de la A-Z en mayúsculas
const patron = /^\d{4}\s?[A-Z]{3}$/;

if (patron.test(document.getElementById("matricula").value)) {
document.getElementById("matricula").classList.remove("error");
```

```
return true;
} else {
alert(
"El campo: Matricula no está correcto.\n\nCuatro números, espacio en blanco opcional y 3 letras mayúsculas."
);
// Situamos el foco en el campo matrícula y le asignamos la clase .error.
document.getElementById("matricula").focus();
document.getElementById("matricula").classList.add("error");
return false;
}
}
}
// Si todos los campos de texto son válidos devolvemos 'true'
return true;
}
//-------------------------------------------------------------//

function validarProvincia() {
// Comprueba que la opción seleccionada sea diferente a 0.
// Si es la 0 es que no ha seleccionado ningún nombre de Provincia.
if (document.getElementById("provincia").selectedIndex == 0) {
alert("Atención!: Debes seleccionar una provincia.");

// Situamos el foco en el campo provincia y le asignamos la clase .error.
document.getElementById("provincia").focus();
document.getElementById("provincia").classList.add("error");
return false;
} else return true;
}

//-------------------------------------------------------------//

// Al hacer click en el botón de enviar se llama a la función 'validar',
// que se encargará de validar el formulario.
document.getElementById("enviar").addEventListener("click", validar, false);
```

Volver