Programación con Javascript

Unidad 7: Formularios y validación

Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 7: Formularios y validación

- Eventos aplicados a formularios
- Validación del lado del cliente

Objetivos

Que el alumno logre:

- Crear y validar formularios con Javascript

Formularios

La programación de aplicaciones que contienen formularios web siempre ha sido una de las tareas fundamentales de JavaScript. De hecho, una de las principales razones por las que se inventó el lenguaje de programación JavaScript fue la necesidad de validar los datos de los formularios directamente en el navegador del usuario. De esta forma, se evitaba recargar la página cuando el usuario cometía errores al rellenar los formularios.

No obstante, la aparición de las aplicaciones AJAX ha relevado al tratamiento de formularios como la principal actividad de JavaScript. Ahora, el principal uso de JavaScript es el de las comunicaciones asíncronas con los servidores y el de la manipulación dinámica de las aplicaciones. De todas formas, el manejo de los formularios sigue siendo un requerimiento imprescindible para cualquier programador de JavaScript.

Propiedades básicas

JavaScript dispone de numerosas propiedades y funciones que facilitan la programación de aplicaciones que manejan formularios. En primer lugar, cuando se carga una página web, el navegador crea automáticamente un array llamado forms y que contiene la referencia a todos los formularios de la página.

Para acceder al array forms, se utiliza el objeto document, por lo que document.forms es el array que contiene todos los formularios de la página. Como se trata de un array, el acceso a cada formulario se realiza con la misma sintaxis de los arrays. La siguiente instrucción accede al primer formulario de la página:

```
document.forms[0];
```

Además del array de formularios, el navegador crea automáticamente un array llamado elements por cada uno de los formularios de la página. Cada array elements contiene la referencia a todos los elementos (cuadros de texto, botones, listas desplegables, etc.) de ese formulario. Utilizando la sintaxis de los arrays, la siguiente instrucción obtiene el primer elemento del primer formulario de la página:

```
document.forms[0].elements[0];
```

La sintaxis de los arrays no siempre es tan concisa. El siguiente ejemplo muestra cómo obtener directamente el último elemento del primer formulario de la página:

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

En un entorno tan cambiante como el diseño web, es muy difícil confiar en que el orden de los formularios se mantenga estable en una página web. Por este motivo, siempre debería evitarse el acceso a los formularios de una página mediante el **array document.forms**.

Una forma de evitar los problemas del método anterior consiste en acceder a los formularios de una página a través de su nombre (atributo name) o a través de su atributo id. El objeto document permite acceder directamente a cualquier formulario mediante su atributo name:

```
var formularioPrincipal = document.formulario;
var formularioSecundario = document.otro_formulario;
<form name="formulario" >
  ...
</form>

<form name="otro_formulario" >
  ...
</form>
```

Accediendo de esta forma a los formularios de la página, el script funciona correctamente aunque se reordenen los formularios o se añadan nuevos formularios a la página. Los elementos de los formularios también se pueden acceder directamente mediante su atributo `name`:

```
var formularioPrincipal = document.formulario;
var primerElemento = document.formulario.elemento;

<form name="formulario">
  <input type="text" name="elemento" />
</form>
```

Obviamente, también se puede acceder a los formularios y a sus elementos utilizando las funciones DOM de acceso directo a los nodos. El siguiente ejemplo utiliza la habitual función document.getElementById() para acceder de forma directa a un formulario y a uno de sus elementos:

```
var formularioPrincipal = document.getElementById("formulario");
var primerElemento = document.getElementById("elemento");

<form name="formulario" id="formulario" >
  <input type="text" name="elemento" id="elemento" />
</form>
```

Independientemente del método utilizado para obtener la referencia a un elemento de formulario, cada elemento dispone de las siguientes propiedades útiles para el desarrollo de las aplicaciones:

- ***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 normales (elemento <select>) su valor es select-one, lo que permite diferenciarlas de las listas que permiten seleccionar varios elementos a la vez y cuyo tipo es select-multiple. Por último, en los elementos de tipo <textarea>, el valor de type es textarea.
- ***form:*** es una referencia directa al formulario al que pertenece el elemento. Así, para acceder al formulario de un elemento, se puede utilizar document.getElementById("id_del_elemento").form
- ***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. Para los campos de texto (<input type="text"> y <textarea>) obtiene el texto que ha escrito el usuario. Para los botones obtiene el texto que se muestra en el botón. Para los elementos checkbox y radiobutton no es muy útil, como se verá más adelante

Eventos aplicados a formularios

Los eventos más utilizados en el manejo de los formularios son los siguientes:

- ***onclick***: evento que se produce cuando se pincha con el ratón sobre un elemento. Normalmente se utiliza con cualquiera de los tipos de botones que permite definir XHTML (<input type="button">, <input type="submit">, <input type="image">).
- ***onchange***: evento que se produce cuando el usuario cambia el valor de un elemento de texto (<input type="text"> o <textarea>). También se produce cuando el usuario selecciona una opción en una lista desplegable (<select>). Sin embargo, el evento sólo se produce si después de realizar el cambio, el usuario pasa al siguiente campo del formulario, lo que técnicamente se conoce como que "el otro campo de formulario ha perdido el foco".
- ***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. Técnicamente, se dice que el elemento anterior "ha perdido el foco"

Validación del lado del cliente

Al crear un formulario en HTML, debemos ser conscientes de un detalle ineludible: los usuarios se equivocan al rellenar un formulario. Ya sea por equivocación del usuario, ambigüedad del formulario, o error del creador del formulario, el caso es que debemos estar preparados y anticiparnos a estos errores, para intentar que los datos lleguen correctamente a su destino y evitar cualquier tipo de moderación o revisión posterior.

Para evitar estos casos, se suele recurrir a un tipo de proceso automático llamado **validación**, en el cuál, establecemos unas pautas para que si el usuario introduce alguna información incorrecta, deba modificarla o en caso contrario no podrá continuar ni enviar el formulario correctamente.

Tradicionalmente, la validación de un formulario se ha hecho siempre en Javascript, sin embargo, HTML5 introduce unos nuevos atributos para formularios que permiten realizar la validación del formulario directamente en HTML5, sin necesidad de recurrir a Javascript.

Atributos básicos de validación

En nuestros campos de entrada de datos, se pueden utilizar ciertos atributos para realizar validaciones sencillas. Algunos de estos atributos ya lo hemos visto en apartados anteriores, sin embargo, vamos a comentarlos uno por uno:

minlength	número	Campos de texto	Establece la longitud mínima del texto requerida.
maxlength	número	Campos de texto	No permite escribir textos superiores a número carácteres.
min	número	Campos numéricos	Establece el número mínimo permitido.
	fecha	Campos de fecha	Establece la fecha mínima permitida.
	hora	Campos de hora	Establece la hora mínima permitida.
max	número	Campos numéricos	Establece el número máximo permitido.
	fecha	Campos de fecha	Establece la fecha máxima permitida.
	hora	Campos de hora	Establece la hora máxima permitida.
step	número	Campos numéricos	Establece el salto de números permitido. Por defecto, 1.
	fecha	Campos de fecha	Establece el salto de días permitido. Por defecto, 1.
	hora	Campos de hora	Establece el salto de segundos permitido. Por defecto, 1.
required		Campos en general	Campo obligatorio. Se debe rellenar para enviar formulario.

disabled		Campos en general	Campo desactivado. No se puede modificar. No se envía.
readonly		Campos en general	Campo de sólo lectura. No se puede modificar. Se envía.

Con estos atributos, podemos crear validaciones básicas en nuestros campos de entrada de datos, obligando al usuario a rellenar un campo obligatorio, forzando a indicar valores entre un rango numérico o permitiendo sólo textos con un tamaño específico, entre otros.

```
<form name="formulario" method="post" action="#">
  <!-- Nombre de usuario. Campo obligatorio, entre 5-40 carácteres -->
 <label> Usuario: </label>
<input type="text" name="nombre" placeholder="Por ejemplo, Juan" minlength="5"
maxlength="40" required>
  <!-- Contraseña. Campo obligatorio, mínimo 11 carácteres -->
  <label>Password: </label>
<input type="password" name="pass" placeholder="Contraseña" minlength="11" required>
</form>
```

También, podemos utilizar las pseudoclases CSS de validación **:valid** e **:invalid** y aplicar estilos a los campos <input> y <textarea> teniendo en cuenta su validación. Aplicando el siguiente código CSS se mostrará un fondo verde o rojo, dependiendo de si cumple o no la validación, respectivamente:

```
input:valid, textarea:valid {
  background:green;
}
input:invalid, textarea:invalid {
  background:red;
}
```

El elemento <input>

El elemento <input> tiene nuevos valores para el atributo type.

```
<input type="email" />
<input type="url" />
<input type="date" />
<input type="time" />
<input type="datetime-local" />
<input type="month" />
<input type="week" />
<input type="number" />
```

```
<input type="range" />
<input type="tel" />
<input type="search" />
<input type="color" />
```

type="email"

Los elementos <input> de tipo "email" permiten al usuario introducir y editar una dirección de correo electrónico, o, si el atributo multiple es especificado, una lista de direcciones de correo electrónico.

```
<input type="email" id="correo" size="30" required>
```

type="url"

El elemento input, teniendo el valor "url" en su atributo type, representa un campo para un solo **URL absoluto**. El control asociado a este campo es una caja de texto que permite a los usuarios editar una sola línea de texto regular.

```
<input type="url" name="sitioweb" placeholder="¿Tienes un sitio web?">
```

type="date"

El elemento input, teniendo el valor "date" en su atributo type, representa un campo para la entrada de una fecha. En los navegadores modernos los campos de fecha son habitualmente representados por controles que permiten a los usuarios cambiar su valor de manera gráfica (como, por ejemplo, un calendario), en lugar de tener que ingresarlo directamente como una cadena.

Previo al envío del formulario, los navegadores que soportan este mecanismo convierten la información ingresada a una cadena que representa una fecha.

```
<input type="date" name="fecha" placeholder="Fecha de nacimiento" >
```

type="datetime-local"

El elemento input, teniendo el valor "datetime-local" en su atributo type, representa un campo para una fecha y hora local sin información de zona horaria. Este tipo de campos es particularmente útil para recolectar un punto en el tiempo, que es específco de un lugar.

Por ejemplo, podría ser utilizado en un formulario para establecer o cambiar la fecha y hora de un simposio, celebrado en una ubicación específica, ya que los participantes deberían encontrarse en el lugar para poder asistir a éste.

En navegadores modernos los campos de fecha y hora global son normalmente representados por controles que permiten a los usarios cambiar su valor de manera gráfica (como, por ejemplo, un calendario), en lugar de tener que ingresarlo directamente como una cadena.

```
<input type="datetime-local" name="horareunion" placeholder="Hora de la reunión" >
```

type="time"

El elemento input, teniendo el valor "time" en su atributo type, representa un campo para la entrada de una hora. En los navegadores modernos, los campos de hora son habitualmente representados por controles que permiten a los usuarios cambiar su valor de manera gráfica, en lugar de tener que ingresarlo directamente como una cadena.

Previo al envío del formulario, los navegadores que soportan este mecanismo convierten la información ingresada a una cadena que representa una hora.

```
<input type="time" name="hora" placeholder="Ingresá la hora" >
```

type="month"

El elemento input, teniendo el valor "month" en su atribtuo type, representa un campo para la entrada de un mes. En los navegadores que soportan el mecanismo, estos campos pueden estar representados por controles que permiten a los usuarios cambiar su valor de manera gráfica (como, por ejemplo, un calendario), en lugar de tener que ingresarlo directamente como una cadena.

```
<input type="month" name="mesnacimiento" placeholder="Mes de nacimiento">
```

type="week"

El elemento input, teniendo el valor "week" en su atributo type, representa un campo para la entrada de una semana. En los navegadores que soportan el mecanismo, estos campos pueden estar representados por controles que permiten a los usuarios cambiar su valor de manera gráfica (como, por ejemplo, un calendario), en lugar de tener que ingresarlo directamente como una cadena.

```
<input type="week" name="unasemana" placeholder="Semana">
```

type="number"

El elemento input, teniendo el valor "number" en su atributo type, representa un campo para la entrada de un número. En los navegadores modernos los campos numéricos son habitualmente

representados por controles que permiten a los usuarios cambiar su valor de manera gráfica, en lugar de tener que ingresarlo directamente como una cadena.

```
<input type="number" name="edad" placeholder="Edad">
```

type="range"

El elemento input, teniendo el valor "range" en su atributo type, representa una campo para la entrada de un número dentro de un rango, con la salvedad de que la elección de un valor exacto no es importante. En los navegadores que soportan el mecanismo los campos de rango son habitualmente representados por algo similar a un deslizador, que se mueve entre los márgenes superior e inferior. Este control es una variante de la entrada numérica que puede desempeñarse mejor en situaciones específicas.

```
<input type="range" name="volumen">
```

type="tel"

El elemento input, teniendo el valor "tel" en su atributo type, representa un campo para un número telefónico. El control asociado a este campo es una caja de texto que permite a los usuarios editar una sola línea de texto regular, sin requerimientos partiuclares sobre el formato. Esta conducta ha sido adoptada debido a la gran variedad de número telefónicos válidos, que hace que las restricciones de propósitos generales sean inapropiadas.

Los autores pueden definir un conjunto particular de restricciones con el atributo pattern.

```
<input type="tel" name="numerocontactado" placeholder="Teléfono">
```

type="search"

El elemento input, teniendo el valor "search" en su atributo type, representa un campo de texto de una sola línea para propósitos de búsqueda. El control asociado a este campo es una caja de texto que permite a los usuarios editar una sola línea de texto regular. Los controles de búsqueda son útiles para recopilar líneas cortas de texto que los agentes procesadores o programas del lado cliente usarán como parámetros para conducir operaciones de búsqueda.

Un campo de texto regular y un campo de búsqueda son muy similares. La diferencia entre ellos es principalemente estilística: los navegadores pueden representar sus controles de manera diferente, de acuerdo a su propósito.

```
<input type="search" name="busquedamusica" placeholder="Canción, autor, álbum...">
```

type="color"

Los elementos **<input>** del tipo «color» proporcionan un elemento de interfaz que permite a los usuarios especificar un color, mediante una interfaz visual de selección o a través de un cuadro de texto donde ingresar un valor hexadecimal en el formato «#rrggbb».Solo se permiten colores simples (sin canal alfa). Los valores son compatibles con CSS.

La presentación del elemento puede variar considerablemente entre navegadores y plataformas: podría ser un campo de entrada sencillo que valida automáticamente que la entrada esté en el formato adecuado, o podría lanzar un selector de colores estándar de la plataforma, o incluso podría abrir una ventana de colores personalizada.

```
<form action="color.php">
  <strong>Selecciona tu color preferido:</strong><br>
  <input type="color" name="favcolor" value="#ff0000">
  <input type="submit">
</form><br>
```

Patrones de validación HTML5

No obstante, aunque los atributos de validación básicos son muy interesantes y pueden facilitarnos la tarea de validación, en muchos casos son insuficientes. Para ello tenemos los patrones de validación HTML5, mucho más potentes y flexibles, que nos permitirán ser mucho más específicos utilizando expresiones regulares para validar datos.

Una expresión regular es una cadena de texto que representa un posible patrón de coincidencias, que aplicaremos mediante el atributo pattern en los campos que queramos validar.

Para ello hay que conocer algunas características básicas de las expresiones regulares:

Expresión regular	Carácter especial	Denominación	Descripción
.	Punto	Comodín	Cualquier caracter (o texto de tamaño 1)
A|B	Pipe	Opciones lógicas	Opciones alternativas (o A o B)

Expresión regular	Carácter especial	Denominación	Descripción
C(A|B)	Paréntesis	Agrupaciones	Agrupaciones alternativas (o CA o CB)
[0-9]	Corchetes	Rangos de carácteres	Un dígito (del 0 al 9)
[A-Z]			Una letra mayúscula de la A a la Z
[^A-Z]	^ en corchetes	Rango de exclusión	Una letra que no sea mayúscula de la A a la Z
[0-9]*	Asterisco	Cierre o clausura	Un dígito repetido 0 ó más veces (vacío incluido)
[0-9]+	Signo más	Cierre positivo	Un dígito repetido 1 ó más veces
[0-9]{3}	Llaves	Coincidencia exacta	Cifra de 3 dígitos (dígito repetido 3 veces)
[0-9]{2,4}		Coincidencia (rango)	Cifra de 2 a 4 dígitos (rep. de 2 a 4 veces)
b?	Interrogación	Carácter opcional	El caracter b puede aparecer o puede que no
\.	Barra invertida	Escape	El caracter literalmente (no como comodín)

Ejemplos de patrones HTML5

Para clarificar estos conceptos y entenderlos mejor, vamos a mostrar algunos ejemplos de campos con validaciones HTML5 en casos reales, y así afianzar conceptos sobre expresiones regulares y su aplicación en los atributos pattern:

- ***Tipo de campo:*** Nombre de usuario
- ***Campo obligatorio:*** required.
- ***Entre 5-40 carácteres:*** minlength="5" maxlength="40"
- ***Sólo se permiten letras (mayúsculas y minúsculas) y números:*** pattern="[A-Za-z0-9]+"

```
<form name="formulario" method="post" action="#">
 <!-- Nombre de usuario. Campo obligatorio, entre 5-40 caracteres. Sólo se permiten letras y números -->
 <input type="text" name="nombre" placeholder="Su nombre de usuario"
     minlength="5" maxlength="40" required pattern="[A-Za-z0-9]+">
</form>
```

Podemos ver que de no incluir los atributos minlength y maxlength el usuario no tendría limitación en cuanto al tamaño. Esto también puede incorporarse en la propia expresión regular, y prescindir de dichos atributos:

```
<form name="formulario" method="post" action="http://pagina.com/send.php">
 <!-- Nombre de usuario. Campo obligatorio, entre 5-40 caracteres. Sólo se permiten letras y números -->
 <input type="text" name="nombre" placeholder="Su nombre de usuario" required pattern="[A-Za-z0-9]{5,40}" title="Letras y números. Tamaño mínimo: 5. Tamaño máximo: 40">
</form>
```

Sin embargo, en este caso, no se limitará al usuario a la hora de escribir, como hace maxlength, sino que permitirá al usuario escribir la información que desee y en caso de no pasar la validación, mostrará un mensaje de advertencia y no lo dejará continuar hasta que termine. Podemos ampliar el mensaje de advertencia incluyendo el texto en el atributo title.

En el siguiente caso, se pide al usuario que indique el modelo de coche que posee, en un posible formulario de servicio técnico. Los modelos posibles son A1, A3, A4 y A15. En lugar de mostrar una lista de selección, podemos mostrar un campo de texto y colocar una validación como la siguiente:

- ***Tipo de campo:*** Modelo de coche
- ***Campo obligatorio:*** required.
- ***Sólo se permiten las opciones:*** A1, A3, A4 y A15

```
<form name="formulario" method="post" action="http://pagina.com/send.php">
  <!-- Modelo de coche. Campo obligatorio, opciones posibles: A1, A3, A4 y A15
      Sólo se permiten letras y números -->
  <input type="text" name="coche" placeholder="Su modelo de coche"
       required pattern="A|a(1|3|4|15)"
       title="Modelos posibles: A1, A3, A4 y A15">
</form>
```

Permite tanto el formato a1 como el formato A1.

Resumen

En esta Unidad…

Trabajamos con Validación de formularios

En la próxima Unidad…

Trabajaremos con Jquery