

# Formularios en JavaScript

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.

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];
```

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>
```

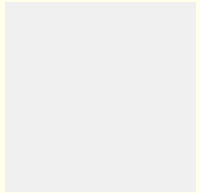
El objeto *form* posee las siguientes propiedades:

propiedad	descripción
<b>name</b>	es el nombre único del formulario.
<b>action</b>	es el lugar al cual se envía el formulario para ser procesado. El <i>action</i> define la URL a la cual se envía dicho formulario.
<b>method</b>	método de envío de los datos insertados en un formulario. El <i>method</i> puede ser:  <b>GET</b> = envía los datos en una cadena "visible". Conveniente para enviar pocos datos.  <b>POST</b> = envía los datos en forma "invisible". Conveniente para enviar una gran cantidad de datos.
<b>target</b>	define la ventana o marco (frame) en la que se mostrarán o procesarán los resultados del formulario.

## Objetos de formulario

Los objetos (elementos) de un formulario son los "campos" de un formulario. En la siguiente tabla, podremos apreciar los tipos de objetos con su correspondiente descripción:

objeto	descripción	ejemplo
text	es un campo de texto en el que los datos en él introducidos son visibles para el usuario.	
<input type="text" name=".." value=".." size="..">		
password	un campo de texto idéntico a <i>text</i> con la diferencia que los datos en él introducidos, no pueden ser visualizados por el usuario, sino que son mostrados con asteriscos *	
<input type="password" name=".." value=".." size="..">		
hidden	es un campo de texto oculto con un valor preestablecido y que el usuario no podrá visualizar en ningún momento.	
<input type="hidden" name=".." value="..">		
textarea	es un campo de texto similar en su tratamiento a <i>text</i> pero lo que en él varía es su apariencia, ya que puede tener un alto y ancho determinado, barras de scroll para navegar por su interior y admite saltos de línea.	
<textarea rows=".." cols="..">....texto...</textarea>		
radio	es un botón circular que permite elegir al usuario una opción de entre varias pertenecientes a un mismo grupo.	
<input type="radio" name=".." value="..">		
checkbox	es una casilla de verificación que permite al usuario seleccionar más de una opción entre varias. Similar a <i>radio</i> pero con esta ventaja.	
<input type="checkbox" name=".." value="..">		
option	es una lista desplegable de varias opciones. Puede permitir la selección de una sola opción o de múltiples opciones (si se mantiene presionada la tecla <b>Ctrl</b> durante dicha selección).	
<select name=".." <b>multiple</b> >		
<option value="..">texto1</option>		
<option value="..">texto2</option>		
</select>		
file	campo compuesto que permite examinar el disco duro para subir ficheros al servidor.	
<input type="file" name=".." size="5">		
submit	es un tipo de botón que se encarga de enviar el formulario.	
<input type="submit" value="..">		

<b>reset</b>	es un tipo de botón que se encarga de restablecer el formulario a sus valores por defecto.	
<code>&lt;input type="reset" value=".."&gt;</code>		
<b>button</b>	es un tipo de botón al que se le pueden asignar múltiples funciones mediante eventos.	
<code>&lt;input type="button" name=".." value=".."&gt;</code>		
<b>image</b>	es una imagen que actúa como botón ( <i>reset</i> , <i>button</i> o <i>submit</i> )	
<code>&lt;input type="image" src=".." ...&gt;</code>		

Independientemente del método utilizado para obtener la referencia a un elemento de formulario, cada elemento dispone de útiles para el desarrollo de las aplicaciones, las mas comunes y utilizadas son:

- **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`.
- **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
- **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`

Los eventos más utilizados en el manejo de los formularios (aunque no sean los únicos que se pueden usar) 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.

- **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 de formularios

La principal utilidad de JavaScript en el manejo de los formularios es la validación de los datos introducidos por los usuarios. Antes de enviar un formulario al servidor, se recomienda validar mediante JavaScript los datos insertados por el usuario. De esta forma, si el usuario ha cometido algún error al rellenar el formulario, se le puede notificar de forma instantánea, sin necesidad de esperar la respuesta del servidor.

Notificar los errores de forma inmediata mediante JavaScript mejora la satisfacción del usuario con la aplicación (lo que técnicamente se conoce como "mejorar la experiencia de usuario") y ayuda a reducir la carga de procesamiento en el servidor.

Normalmente, la validación de un formulario consiste en llamar a una función de validación cuando el usuario pulsa sobre el botón de envío del formulario. En esta función, se comprueban si los valores que ha introducido el usuario cumplen las restricciones impuestas por la aplicación.

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.

```
<form action="" method="" id="" name="" onsubmit="return validacion()">
...
</form>
```

```
function validacion() {
  if (condicion que debe cumplir el primer campo del formulario) {
    // Si no se cumple la condicion...
    console.log('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  else if (condicion que debe cumplir el segundo campo del formulario) {
    // Si no se cumple la condicion...
    console.log('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  // Si el script ha llegado a este punto, todas las condiciones
  // se han cumplido, por lo que se devuelve el valor true
  return true;
}
```

De este modo podemos validar los campos antes de enviar la información del formulario pero también nos puede interesar ir validando la información que nos suministran en tiempo real fijando restricciones a la hora de introducir la información en los textareas, cuadros de texto, cuadros de password, etc.

En cualquiera de los casos, validar los datos de un formulario es algo muy habitual para tareas como:

- Asegurarse de que en un campo numérico no se escriben letras.
- Asegurarse de que dos campos password son idénticos.
- Asegurarse de marcar una opción de una lista desplegable.
- Asegurarse de no dejar sin marcar ninguna casilla de verificación.
- Asegurarse de escribir bien el formato de email.
- Asegurarse de escribir bien en un formato de fecha determinado.