MEDAC
Instituto Oficial de Formación Profesional

DESARROLLO WEB EN ENTORNO CLIENTE
TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Interacción con el usuario

08

ÍNDICE

OBJETIVOS

Saber utilizar formularios y sus atributos.

Conocer los elementos de los formularios.

Entender los mecanismos de envío del formulario.

Conocer diferentes métodos de validación.

Entender la ejecución de eventos en un formulario.

Saber crear expresiones regulares.

/ 1. Introducción y contextualización práctica

Uno de los aspectos más importantes a la hora de interactuar con el usuario es la recepción y manipulación de los datos que hay que enviar al servidor. HTML ofrece el elemento «formulario» para permitir al usuario introducirlos.

En este sentido, HTML ofrece diferentes tipos de elementos que se pueden introducir en un formulario. Gracias a ello, el usuario podrá elegir, por ejemplo, entre un cuadro de texto en claro o un texto enmascarado. Actualmente existe un gran número de funcionalidades que se pueden implementar a través de la web. Por ello, es importante crear páginas web con interfaces de usuario que sean fáciles de usar y que permitan la interacción con el usuario de manera similar a las herramientas de sobremesa. Los formularios se convierten, así, en una herramienta de interacción con el usuario de nuestra web.

Fig. 1. La Interacción con el usuario puede marcar la diferencia

Escucha el siguiente audio que describe el caso práctico que iremos resolviendo a lo largo de la unidad:

Audio Intro. "Validación de formularios"
https://bit.ly/3hJaLE0

/ 2. Formularios en HTML

Hoy en día es muy común que una página web ofrezca ciertas capacidades para poder interactuar con el usuario. La forma más sencilla para poder interactuar él es mediante los formularios.

Estos pueden ser manejados desde JavaScript, pero tienen que ser previamente definidos en el documento web. Los formularios permiten introducir datos que pueden ser enviados directamente al servido, que se encargará de procesar los datos y de generar la respuesta correspondiente a lo que se haya solicitado o enviado para su procesamiento. Para crear un formulario en HTML se utiliza la etiqueta <*form*>.

2.1. Elementos de los formularios

Un formulario puede contener diferentes tipos de elementos. A continuación, vamos a ver los más importantes para poder crear una web que facilite la interacción con el usuario.

Elemento	Descripción
<input>	Permite introducir un valor
<select>	Permite realizar una selección
<textarea>	Visualiza un área de texto
<button>	Permite crear un botón
<fieldset>	Establecer una relación entre los datos del formulario
<datalist>	Establece valores definidos en <input>
<option>	Crea una opción en una lista desplegable

Tabla 1. Elementos HTML para los formularios

Combinando los elementos del formulario junto con sus atributos, podemos crear una representación visual que satisfaga las necesidades de los usuarios. Gracias a los atributos se puede cambiar significativamente el comportamiento de un elemento del formulario. El conjunto de atributos viene definido en la documentación oficial de HTML.

2.2. Ejemplos de formularios en HTML

Veamos algún ejemplo de formularios en HTML.

Este formulario, por ejemplo, permite crear dos campos de entrada para que el usuario pueda introducir su nombre y apellidos:

```
<form>
  <label for="fname">Nombre:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Apellidos:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

Código 1. Formulario HTML para introducir nombre y apellidos

En este otro ejemplo, permitimos mostrar botones de selección en los que se puede seleccionar solo una: perro o gato.

```
<form>
 <input type="radio" id="dog" name="pet" value="dog">
 <label for="dog">Perro</label><br>
 <input type="radio" id="cat" name="pet" value="cat">
 <label for="cat">Gato</label><br>
</form>
```

Código 2. Formulario HTML con botones de selección

2.3. Tipos de elementos <input>

El elemento <*input*> es quizás el elemento más utilizado dentro de un formulario HTML. Además, es el más versátil, pues su comportamiento se adapta fácilmente, lo cual permite cambiar tanto la representación visual como el tipo de dato admitido. Todo ello se logra utilizando el atributo *type*, que puede tomar diferente valores como *button, checkbox, color, date, email, file*, etc. Dependiendo del valor de este atributo, la representación de este campo cambiará. Por ejemplo, si queremos representar un cuadro de texto utilizamos el valor *text*. En el caso de querer representar la entrada de una contraseña, utilizamos el valor *password*. En el primer caso, se muestra un texto en claro, mientras que en el segundo se muestra un texto enmascarado.
Veamos en el siguiente vídeo más casos de procesamiento de formularios:

Vídeo 1. "Procesamiento de formularios con JavaScript"
https://bit.ly/2Bo4QmY

2.4. Otros aspectos sobre <input>

Existe una diferencia entre un botón y un campo de envío. La representación visual de ambos es exactamente la misma, específicamente, un botón. Sin embargo, la definición y el significado se hace de manera diferente. Veamos un ejemplo:

```
<!--Este formulario permite enviar valores al servidor, pues al ejecutar
el submit se ejecutará la acción del formulario-->
<form action="/action_host">
 <label for="fname">Nombre:</label><br>
 <input type="text" id="fname" name="fname"><br>
 <label for="lname">Apellidos:</label><br>
 <input type="text" id="lname" name="lname">
 <input type="submit" value="Enviar">
</form>
```

Código 3. Envío de valores al servidor

HTML también ofrece la posibilidad de crear fácilmente la funcionalidad de restablecer el contenido de un formulario a su valor por defecto, incluyendo *type="reset"*.

Otros comportamientos para este elemento se definen a continuación:

type=	Descripción
"radio"	Define un botón de radio
"checkbox"	Permite definir una caja de muchas opciones
"button"	Permite definir Botón

Tabla 2. Valores del atributo type de los tags input en HTML

Este elemento permite que el usuario introduzca manualmente los datos y existen mecanismos para introducir limitaciones. Utilizando atributos podemos modificar el comportamiento, por ejemplo, fijando la longitud máxima y mínima que debe tener la entrada del usuario.

/ 3. Caso práctico 1: "Sin comunicación con el servidor"

Planteamiento: En nuestra empresa, el equipo de web adaptativa nos comenta que el siguiente código no comunica con el servidor:

```
<form action="server.php">
<button>Enviar</button>
</form>
```

Código 4. Error de comunicación

Nudo: ¿Qué crees que puede estar pasando? ¿Es correcto este código?

Desenlace: Como hemos visto en el apartado sobre la creación de formularios HTML, a través de la etiqueta <*button*> no se envía un formulario. Para ello es necesario utilizar una etiqueta <*input*> con el tipo *submit*. A través de este elemento, se ejecutará la acción marcada por el formulario. En el caso del Código 4, sería tan sencillo como cambiar la etiqueta <*button*> por <input>.

/ 4. Atributos y envíos de un formulario

4.1. Atributos de un formulario

Al igual que sucede con los elementos de un formulario, el propio formulario puede tener atributos asociados que permiten definir su comportamiento cuando ocurre un tipo de evento, es decir, para reaccionar ante determinadas circunstancias.

Los formularios pueden tener un identificador que se puede asociar a un campo de tipo *input*, de tal forma que dicho campo aunque esté declarado fuera del propio formulario, queda vinculado a él. El formulario puede definir, a través del atributo *action*, qué tarea debe llevar a cabo cuando se envía al servidor.

Este comportamiento puede ser modificado utilizando el atributo *form action* asociado a un elemento de tipo <*input*>.

Otro campo importante de los formularios es el método de envío que puede tomar los valores *POST* o *GET*. El método de envío fijará si los parámetros del formulario se envían a través de la URL o bien a través del cuerpo de la petición HTTP. Revisemos algunas consideraciones que hay que tener en cuenta acerca del método de envío.

4.2. Método de envío GET

Este método une los datos del formulario a la URL en formato clave-valor. Los nombres de los campos van a coincidir con los nombres del campo en el formulario. Este método de envío es útil cuando el usuario desea obtener un resultado. Hay que tener en cuenta que existe un límite sobre la cantidad de datos que se puede incluir en una URL y que dependerá del navegador que estemos utilizando. Es importante que no se use este método para el envío de información sensible como, por ejemplo, contraseñas, porque pueden ser visualizada directamente en la barra de navegación del navegador.

4.3. Método de envío POST

Este método es capaz de enviar los datos del formulario usando una petición POST HTTP. A diferencia del método anterior, no tiene limitación en el tamaño de datos enviados y puede ser más robusto a la hora enviar datos sensibles. En general, ninguna información sensible debería ser enviada sin encriptar.

4.4. Validación y envío de formularios

La validación de formularios HTML puede ser realizada desde JavaScript y HTML. Una vez que hemos visto cómo acceder a los elementos del formulario, podemos realizar su validación.

La validación de formulario es una parte específica de la validación de datos que garantiza que una aplicación tenga un comportamiento adecuado, sobre todo cuando interactúa con el usuario. Al estar desarrollando en el cliente, toda la interacción con el usuario sucede de manera directa, por ello es importante realizar la validación en este punto.

La validación de datos se puede realizar de diferentes formas y en diferentes momentos de la arquitectura cliente/servidor. A grandes rasgos, podemos distinguir dos tipos de validaciones:

- **Validación en cliente**. Antes de enviar cualquier dato al servidor se realiza una comprobación de los datos.
- **Validación en servidor**. En este tipo de validación el navegador no realizar ninguna comprobación y todo se realiza en el servidor. El problema de esta solución es que se puede sobrecargar la red y el servidor.

Fig. 2. Validando formularios en HTML

/ 5. Objetos de un formulario

Una vez hemos observado cómo generar formularios en HTML, podemos realizar su manipulación y control a través de JavaScript. Para ello, en primer lugar, necesitamos acceder al objeto del formulario a través del DOM. Un requisito fundamental es que todo formulario al que queramos acceder deberá disponer de un identificador. No solo es posible acceder a un formulario a través de JavaScript, sino que también podemos crear un formulario de manera dinámica.

5.1. Validación desde HTML

La validación de un formulario a través de HTML puede desarrollarse automáticamente por el navegador. Este tipo de comprobaciones suelen ser bastante sencillas como, por ejemplo, que un campo no haya sido introducido. Otras funcionalidades de la validación HTML implican comprobar la longitud de los elementos de entrada cuando el usuario tiene que teclear ciertos valores. Otro escenario es que se tenga que comprobar el tipo de dato que se ha introducido o bien si la entrada se ajusta a determinados patrones. Con la introducción de HTML5, la validación se basa en la satisfacción de determinadas restricciones.

5.2. Eventos de un formulario

Los eventos forman parte de la programación dirigida por eventos, que se fundamenta en realizar determinadas tareas cuando tiene lugar un cambio en el entorno, es decir, un evento. Se trata de un tipo de programación que está muy relacionada con el diseño de interfaces de usuario. HTML ofrece la posibilidad de añadir funciones JavaScript cuando ocurre un evento sobre prácticamente cualquier elemento de la web. En el caso de los formularios y de los elementos que lo componen, es posible responder a los diferentes eventos que pueden tener lugar. Algunos de estos son:

Evento	Descripción
onchange	Se lanza al cambiar un elemento
oninput	Se lanzan cuando se introduce un valor
onreset	Se lanza al restablecer el formulario
onselect	Se lanza al seleccionar un elemento
onsubmit	Se lanza al enviar el formulario

Tabla 3. Eventos HTML para acciones JavaScript

5.3. Validación desde JavaScript

La validación mediante JavaScript requiere de la utilización de eventos y de código JavaScript que realice la correspondiente validación. A diferencia de la validación en HTML, al utilizar JavaScript, la validación no es automática. Por este motivo, está basado en eventos. La forma más sencilla de realizar la validación es a través de la definición de funciones que podamos vincular a los eventos sobre un formulario. A continuación, verás en un ejemplo la validación de formulario utilizando JavaScript.

A través del DOM se podrán realizar la comprobación de los campos de un formulario. Para ello se ofrecen diferentes propiedades que pueden ser invocadas en la definición de las funciones de comprobación cuando se envía el formulario. Por ejemplo:

```
function checkForm() {
  var x = document.forms[“testForm”][“fname”].value;
  if (x == “”) {
    alert(“Por favor, introduce un nombre.”);
    return false;
  }
}
<form name=”testForm” action=”/action_page.php”>
  <label for=”fname”>First name:</label><br>
  <input type=”text” id=”fname” name=”fname” value=”John”><br>
  <label for=”lname”>Last name:</label><br>
  <input type=”text” id=”lname” name=”lname” value=”Doe”><br><br>
  <input type=”submit” value=”Submit” onsubmit=”return checkForm()”>
</form>
```

Código 5. Comprobaciones de los campos de un formulario

Vídeo 2. “Validación de formularios con JavaScript”
https://bit.ly/3hJ8Qzi

5.4. Expresiones regulares

Una expresión regular permite definir el aspecto y los caracteres admitidos en un identificador o valor de entrada. Existen diferentes opciones para representar las expresiones regulares y en HTML se utiliza una de ellas. La expresión regular no se utiliza únicamente en entornos web, sino que puede ser utilizada en cualquier contexto de programación.

Para crear una expresión regular hay que determinar el conjunto de caracteres y cómo se pueden combinar para permitir la creación de representaciones válidas.

En HTML las expresiones regulares se utilizan cuando se quieren comprobar los valores de entrada aportados por los usuarios. Como los valores se pueden introducir utilizando elementos del tipo <*input*>, se añade un atributo llamado *pattern*, al que se le asigna la expresión regular.

A continuación, se muestra un ejemplo de validación mediante expresiones regulares:

```
<!--Este formulario permite crear dos campos de entrada que el usuario pueda introducir el nombre y los apellidos. El nombre incluye una expresión regular para solo permitir minúsculas y mayúsculas-->
<form>
 <label for="fname">Nombre:</label><br>
 <input type="text" id="fname" name="fname" pattern="[A-Za-z]{20}"><br>
 <label for="lname">Apellidos:</label><br>
 <input type="text" id="lname" name="lname">
</form>
```

Código 6. Validación mediante expresiones regulares

Audio 1. "Expresiones regulares"
https://bit.ly/2UZmqos

/ 6. Caso práctico 2: "Securizar la información"

Planteamiento: En un proyecto para la creación de formularios con un amplio número de campos, es necesario extremar las medidas de seguridad. El cliente nos pide que todo el flujo de la comunicación sea lo más seguro posible, dada la cantidad de datos que van a viajar.

Nudo: ¿Cómo podrías realizar esto? ¿Qué tendrías que responder al cliente?

Desenlace: Como hemos visto en el apartado sobre la validación de formularios, es posible realizar la validación en el cliente utilizando JavaScript y HTML. Además, esto es recomendable para poder liberar al servidor y la red de tráfico innecesario. Sin embargo, el cliente nos pide que incluyamos medidas de seguridad redundantes. En este caso, podemos realizar comprobaciones en el servidor para confirmar que los datos siguen siendo válidos después de viajar por la red. Es posible utilizar HTTPS para la comunicación para evitar que los datos puedan ser capturados.

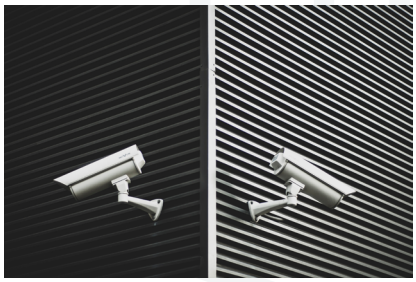

Fig. 3. La seguridad es un aspecto fundamental en el mundo web

/ 7. Resumen y resolución del caso práctico de la unidad

En este tema hemos presentado los principales aspectos para **manipular formularios tanto desde HTML como desde JavaScript**. Como hemos visto a lo largo de la unidad, **para poder manipular los formularios desde JavaScript es necesario que estos existan previamente en el documento web**. Además, un formulario debe poseer un identificador para poder manipularlo desde JavaScript.

En HTML los formularios pueden tener diferentes elementos en su interior, los cuales deben estar asociados a un identificador válido para que pueda ser recuperado correctamente desde JavaScript. Los formularios pueden ser validados tanto a nivel de HTML como utilizando funciones propias de JavaScript. En este último caso, es necesario introducir eventos en el formulario para poder responder, al menos, al evento de envío.

Hemos comprobado, también, que HTML ofrece otros métodos de validación de un formulario, por ejemplo, utilizando expresiones regulares.

En este tema **también hemos observado cómo los elementos de un formulario pueden cambiar su comportamiento utilizando atributos** dentro de sus etiquetas.

Resolución del caso práctico de la unidad

Como hemos visto a lo largo del tema, la utilización de expresiones regulares puede usarse tanto en HTML como en JavaScript. Sin embargo, la validación en JavaScript permite realizar validaciones más complejas basadas en otros campos como el código postal. Realizando la validación en el cliente evitamos que el servidor se sature realizando comprobaciones que pueden ser delegadas en los clientes.

/ 8. Bibliografía

Pilgrim, M. (2010). HTML5: *up and running*. Sebastopol: O'Reilly Media, Inc.

Caso. (2020). *Curso de JavaScript*. City: Síntesis.

Gómez, M. (2017). *Curso de Desarrollo Web:* HTML, CSS y JavaScript. Madrid: Anaya Multimedia.