ThePower business school

Etiquetas formularios HTML⁉️👨‍💻

Los formularios son los elementos en los que se integran los botones y las áreas de texto que se utilizan para que los usuarios introduzcan sus datos o que puedan elegir entre varias opciones. Veamos las etiquetas que se utilizan para la creación de formularios en HTML5.

Creación de un formulario

Los formularios son las piezas **HTML5** que nos permiten recoger la información que introduce el usuario. Hay algunos elementos que son básicos dentro de nuestro formulario.

Creación de un formulario: Etiqueta form

Siempre usamos la etiqueta **<form>** para englobar o encerrar todo el contenido del formulario. En esta etiqueta se suelen definir tres atributos:

1) **action** , que sirve para definir la localización donde se enviará la información que se recoja en el formulario, y

2) **method, que lo usaremos** para definir el método HTTP en el que se enviará esta información en la que puede ser **get** o **post.**

3) name, que hace referencia al nombre del formulario.

Os dejamos un sencillo ejemplo de código:

```
<form name="mi formulario" action="/next-page" method="post"></form>
```

Otros atributos que puede recibir esta etiqueta son:

Atributo	Valor	Descripción
target	destino	Nombre del lugar donde se abrirá el formulario. _blank para nueva pestaña.
enctype	application/x-www-form-urlencoded, multipart/form-data, text/plain	Codificación para el envío del formulario. Importante para envío de archivos.

```
<form action="/action.php" target="_blank" accept-charset="utf-8">
  <label for="fname">First name:</label>
```

```
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
</form>
```

```
<form action="/action.asp" method="post" novalidate enctype="multipart/form-data">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
</form>
```

```
<form action="/action.php" method="get" autocomplete="on">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="email">Email:</label>
  <input type="text" id="email" name="email"><br><br>
  <input type="submit">
</form>
```

Creación de un formulario: Etiqueta input

La etiqueta **<input>** es la etiqueta más versátil, define un campo dentro de un formulario y tiene tres atributos principales :

La etiqueta `<input>` especifica un campo de entrada donde el usuario puede ingresar datos.

Esta etiqueta es la más importante de un formulario, y se podrá mostrar de diferentes formas en función de sus types:

El atributo **type** define el comportamiento que tendrá el input por defecto. Algunos de los tipos de inputs son: **text**, **checkbox**, **date**, **number**, **password**, **reset**, **radio**, **submit**, **file**...

Como veis hay muchos tipos, os dejamos a continuación todos ellos para que los tengáis presentes:

```
<input type="button">
<input type="checkbox">
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="email">
<input type="file">
<input type="hidden">
<input type="image">
<input type="month">
<input type="number">
```

```
<input type="password">
<input type="radio">
<input type="range">
<input type="reset">
<input type="search">
<input type="submit">
<input type="tel">
<input type="text"> (por defecto)
<input type="time">
<input type="url">
<input type="week">
```

El atributo `name` especifica un nombre a cada uno de los `<input>` del formulario. Además, se utiliza para hacer referencia a elementos de JavaScript o para hacer referencia a datos de formulario después de enviar un formulario, por lo que este atributo nos ayudará a la hora de enviar la información recogida al backend.

> **Nota:** solo los formularios que contengan este atributo enviarán información.

```
<label for="fname">First name:</label>
<input type="text" id="fname" name="fname">
<label for="lname">Last name:</label>
<input type="text" id="lname" name="lname">
<input type="submit" value="Submit">
```

Creación de un formulario: Qué hemos aprendido

1. Cómo crear un formulario en nuestra página web
2. Los atributos que acompañan a los formularios.
3. Los tipos de datos que se pueden insertar en formularios.
4. Qué son y cómo usar los inputs.

Campos de texto

La etiqueta <input> puede tener diferentes types, entre los que se encuentran los de tipo texto, que serán:

```
<input type="email"> el teclado mostrará botones especiales como .com y @
<input type="hidden">info que queremos enviar con el formulario pero que no se vea en la pantalla
<input type="password"> no aparecerán los caracteres visibles al escribir.
<input type="search"> para búsquedas
<input type="tel"> tipo teléfono, desplegará teclado numérico
<input type="text"> (por defecto)
<input type="url">
<textarea>
```

A estos inputs les pueden acompañar los atributos autocomplete, que indica si queremos que aparezca información anteriormente escrita en este campo y placeholder, que será un texto a modo de sugerencia/ayuda para el usuario.

Existe además uno complementario que es spellcheck para que el navegador compruebe la ortografía cuando escribimos.

Campos de texto: Input de tipo text

El campo de texto, como bien indica su nombre, es el que nos permite insertar texto en un formulario. El usuario podrá insertar texto visible sobre el campo.

```
<input type="text" id="identificador" size="tamaño"
  name="nombre" value="texto por defecto"/>
```

De esta forma, si queremos crear un campo de texto para poder insertar 70 caracteres que contenga un email, lo definiremos de la siguiente forma.

```
<input type="text" id="email" name="email"
  size="70" value="usuario@dominio.com"/>
```

Campos de texto: Input de tipo password

Cuando un usuario inserte una contraseña dentro de un formulario necesitaremos, casi seguro, que el valor de la contraseña no aparezca y que por el contrario aparezcan caracteres como asteriscos.

```
<input type="password" id="identificador" size="tamaño" name="nombre"/>
```

Si queremos definir un campo que acepte contraseñas de 20 caracteres lo codificaremos de la siguiente forma:

```
<input type="password" id="clave" name="clave" size="20"/>
```

Campos de texto: Etiqueta textarea

Además de los inputs tenemos la etiqueta **<textarea>** que es muy similar al input con el **type=”text ”** con la diferencia que no se autocierra y nos permite recoger textos con mayor volumen, como son los comentarios,

descripciones, observaciones… Esta etiqueta también puede utilizar el atributo **id** para enlazarse con un **<label>.**

La etiqueta `<textarea>` define un control de entrada de texto de varias líneas. Además, se usa a menudo en un formulario, para recopilar las entradas de los usuarios, como comentarios o reseñas.

Como curiosidad, comentar que un <textarea> puede contener una cantidad ilimitada de caracteres y el texto se muestra por defecto en una fuente de ancho fijo (generalmente Courier).

El atributo `name` es necesario para hacer referencia a los datos del formulario después de enviar el formulario. Asimismo, el atributo `id` es necesario para asociar el área de texto con una <label>.

Como siempre os dejamos un ejemplo de código para mayor facilidad:

```
<label for="comments">Comentarios:</label>
<textarea id="comments">Este será el contenido de los comentarios</textarea>
```

Campos de texto: Algunos de los atributos más usados son

minlength y **maxlength** en los input de tipo **text** son usados para indicar un tamaño mínimo o máximo de caracteres en una entrada de datos. Así podremos restringir un input para que al menos tenga **n** caracteres, o que no se exceda de **n** caracteres.

En el ejemplo estamos diciendo que el input de tipo texto referente al “nick” debe ser mínimo de 2 caracteres de largo y máximo 8.

```
<form method="post" action="/user">
  <label for="nick">nickName:</label>
  <input type="text" name="nick" placeholder="Su nick" minlength="2" maxlength="8">
  <button type="submit">Enviar</button>
</form>
```

Campos de texto: Qué hemos aprendido

1. Los tipos de inputs de texto que existen.
2. Los atributos que los acompañan.
3. Cómo usar textareas.

Campos numéricos

Si deseamos obtener información o cantidades numéricas en un formulario, tenemos a nuestra disposición dos tipos de etiquetas `<input>`

```
<input type="range">
<input type="number">
```

min y **max** para los **inputs** de type **number** que igual manera que con los campos de texto, podemos restringir los campos números mediante un valor máximo y un valor mínimo.

En este ejemplo, la edad que debemos rellenar podrá ir de los 14 a los 99 años.

```
<form method="post" action="/user">
    <label for="age">Edad:</label>
    <input type="number" name="age" placeholder="Su edad" min="14" max="99">
  <button type="submit">Enviar</button>
</form>
```

Campos numéricos: Atributo step

El atributo `step` especifica los intervalos de números legales para un campo de entrada.

Ejemplo: si el valor de step es = "5", los números válidos serían -5, 0, 5, 10, etc.

Este atributo se puede usar junto con los atributos max y min que vimos más arriba para crear una validación más sofisticada.

Este atributo funciona con los siguientes tipos: número, rango, fecha, fecha y hora local, mes, hora y semana.

```
<form>
  <label for="points">Points:</label>
  <input type="number" id="points" name="points" step="5">
</form>
```

Campos numéricos: Qué hemos aprendido

1. Los tipos de inputs para números que existen y cómo usarlos.
2. Los atributos más comunes que acompañan a este tipo de input.

Campos date & hour

Si nuestra intención es que el usuario introduzca una fecha concreta, en lugar de utilizar un campo de texto, lo ideal sería utilizar un control llamado datepicker, que es básicamente un calendario incorporado en HTML que se le abrirá al usuario. Existe lo mismo para las horas, en este caso el timepicker. Los tipos de inputs de fecha y hora son:

```
<input type="date">
<input type="datetime-local">
<input type="month">
<input type="time">
<input type="week">
```

min y **max** para los **inputs** de type **fecha y hora son aptos de** igual manera que con los campos de texto, podemos restringir los campos números mediante un valor máximo y un valor mínimo. Igualmente se puede usar el atributo step.

```
<form name="formulario" method="post" action="/send.php">
  <!-- Campo de entrada de fecha -->
  Selecciona la fecha deseada:
  <input type="date" name="fecha" min="2034-03-25" max="2038-05-25" step="2" />
  <!-- Campo de entrada de hora -->
  Selecciona la hora deseada:
  <input type="time" name="hora" min="18:00" max="21:00" step="3600" />
</form>
```

Campos date & hour: Qué hemos aprendido.

1. Los tipos de campos de fecha y hora que existen y cómo usarlos.
2. Los atributos que se les pueden añadir a estos tipos.

Botones | check | radios

Mediante los botones podremos realizar operaciones de envío del formulario, de manipulación de datos, borrado,…

Existen dos formas de insertar botones dentro de un formulario: el elemento input y el elemento button. La sintaxis para un botón mediante un elemento input será, como vimos en la anterior sección:

```
<input type="button" value="TextoBotón"/>
```

Botones | check | radios: Input checkbox

También tenemos los **checkbox** nos permite capturar un dato del usuario mediante un elemento de check. El **checkbox** puede tener dos valores, seleccionado o no seleccionado.

```
<input type="checkbox" id="identificador" name="nombre"/>
```

Podemos hacer es generar un checkbox que esta preseleccionado. Para ello utilizamos el atributo **checked**.

```
<input type="checkbox" id="identificador" name="nombre" checked="checked"/>
```

Si os preguntáis para qué podemos usar un **checkbox** tenemos el ejemplo claro de unas condiciones y podríamos representarlo de la siguiente manera.

```
<input type="checkbox" id="condiciones" name="condiciones"/>Está de acuerdo con las condiciones explicadas más arriba.
```

Los checkbox suelen ir en grupos para seleccionar varias opciones. Por ejemplo podríamos tener el siguiente código con el que podamos seleccionar qué lenguaje de programación queremos aprender.

```
<input type="checkbox" name="lenguaje" value="html">HTML
<input type="checkbox" name="lenguaje" value="javascript">Javascript
<input type="checkbox" name="lenguaje" value="css">CSS
<input type="checkbox" name="lenguaje" value="xml">XML
```

Botones | check | radios: Input radio

Con los elementos de radio podemos ofrecer un conjunto de opciones al usuario de tal manera que solo pueda elegir una de ellas. La sintaxis que seguiremos en los elementos input de tipo radio será la siguiente:

```
<input type="radio" id="identificador" value="valor" name="nombre"/>
```

En el caso de los elementos radio toma un papel principal el atributo **name**. Ya que para poder agrupar opciones todos tendrán que tener el mismo valor de atributo name.

Así, si queremos crear un grupo de radios para que nos elija una edad le podremos crear de la siguiente forma:

```
<input type="radio" id="menos18" value="menos18" name="edad"/>Menos de 18
<input type="radio" id="18a30" value="18a30" name="edad"/>18 a 30
<input type="radio" id="31a50" value="31a50" name="edad"/>31 a 50
<input type="radio" id="mas50" value="mas50" name="edad"/>Más de 50
```

Al igual que sucedía con los campos de entrada de tipo check, podemos marcar por defecto el elemento radio mediante el atributo checked.

```
<input type="radio" id="identificador" value="valor" name="nombre" checked="checked"/>
```

Botones | check | radios: Qué hemos aprendido

1. Qué son y para qué sirven los botones.
2. Los de tipo Radio, qué son, cómo se usan y sus atributos.
3. Los de tipo Checkbox, qué son, cómo se usan y sus atributos.

Listas de selección

Si necesitamos mostrar una lista extensa de datos tenemos a nuestra disposición elementos de listas, que permiten al usuario ver varias opciones pero que obligará a que elija únicamente una.

Listas de selección: Etiqueta select

La etiqueta `<select>` se utiliza para crear una lista desplegable. Se usa con mayor frecuencia en un formulario, para recopilar información obtenida de la selección del usuario.

El atributo `name` es necesario para enlazar el select con el resto del formulario, es decir, es necesario para que la información recogida se envíe correctamente (si no lo ponemos no se enviará ningún dato).

El atributo `id` es necesario para asociar la lista desplegable con una etiqueta.

Dentro de la etiqueta <select> habrá diferentes etiquetas <option>, que contendrán las posibilidades que el usuario tiene para elegir.

Es importante recalcar que en un <select> el usuario podrá escoger únicamente una opción. Si queremos dejar que el usuario pueda seleccionar varias opciones tendremos otros elementos disponibles.

Para redondear un elemento <select> podemos añadirle encima una etiqueta <label> que será lo que definirá la etiqueta o “título” del select que le relacionemos.

```
<label for="avengers">Elige una opción:</label>

<select name="avengers" id="avengers">
  <option value="thor">Thor</option>
  <option value="hulk">Hulk</option>
  <option value="ironman">Ironman</option>
</select>
```

Listas de selección: Etiqueta optgroup

Otra de las cosas que podemos realizar dentro de un combo es agrupar opciones. Si la lista de opciones es muy grande podemos utilizar el elemento **<optgroup>**. La sintaxis del elemento **<optgroup>** es la siguiente:

```
<optgroup label="etiqueta"></optgroup>
```

Vamos a verlo con un ejemplo sencillo y así veis a lo que hacemos referencia.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="./styles.css">
  <title>Footer</title>
</head>

<body>
    <select name="ciudad">
        <optgroup label="Europa">
          <option>Madrid</option>
          <option>Londres</option>
          <option>Paris</option>
        </optgroup>
        <optgroup label="Suramerica">
          <option>Santiago</option>
          <option>Sao Paulo</option>
          <option>Lima</option>
          <option>Bogota</option>
        </optgroup>
          <optgroup label="Africa">
          <option>Casablanca</option>
          <option>Ciudad del Cabo</option>
        </optgroup>
      </select>

</body>

</html>
```

Como vemos, dentro del <select> habrá grupos (Europa, Sudamérica y África) y, dentro de esos grupos, habrá distintas opciones.

Si no lo habéis entendido del todo os animamos a que pongáis ese código en vscode y lo visualicéis en el navegador!

Listas de selección: Etiqueta datalist

Con esta etiqueta crearemos una lista abierta, donde el usuario puede seleccionar una opción de las sugeridas o crear la suya propia. Dentro de este <datalist> habrá diferentes <option>.

Visualmente el datalist no mostrará nada, sin embargo, le asignaremos un input a través del atributo id y esto nos posibilitará escribir como si se tratase de un input de type text normal y además tendremos las opciones para elegir en el desplegable.

Os dejamos un ejemplo para mejor entendimiento:

```
<form name="formulario" method="post" action="/send.php">
  <!-- Campo de texto combinado con lista de opciones -->
  <input type="text" list="items" />
  <!-- Lista de opciones -->
  <datalist id="items">
    <option value="1">Opción 1</option>
    <option value="2">Opción 2</option>
    <option value="3">Opción 3</option>
  </datalist>
</form>
```

Con lo que tendremos que tener cuidado es con el soporte que este elemento tiene en los distintos navegadores, ya que nos podemos encontrar alguno que no lo tenga y este elemento no se muestre correctamente al correr nuestra aplicación.

Listas de selección: Qué hemos aprendido

1. Qué son las listas de selección y qué opciones tenemos para crearlas.
2. Los atributos que las acompañan.
3. Qué es un datalist y por qué nos puede resultar útil usarlo.

Selección de color

Con HTML5 se introdujo un nuevo campo de entrada de datos que se usa para seleccionar colores. Este input de tipo color se usa generalmente con el esquema de colores RGB.

```
<form name="formulario" method="post" action="http://mipagina/send.php">
  <!-- Selección de color -->
  Selecciona el color deseado:
  <input type="color" name="color" value="#FF0000" />
</form>
```

Selección de color: Qué hemos aprendido

1. Cómo se usa el input de tipo color y para qué sirve.

Selección de ficheros

HTML nos proporciona un input de tipo file para todo lo que tiene que ver con archivos. Podremos seleccionar, adjuntar y enviar archivos con este input.

Selección de ficheros: Input file

Cuando diseñemos un formulario es posible que necesitemos enviar un fichero de datos al servidor. En este caso vamos a necesitar un campo de entrada de tipo **file**. La sintaxis de este tipo de input sería:

```
<input type="file" id="identificador" value="valor" name="nombre"/>
```

Selección de ficheros: Atributo múltiple

El atributo `multiple` especifica que el usuario puede ingresar más de un valor en un campo. Tenéis que tener en cuenta que solo funciona con los siguientes tipos de entrada: correo electrónico y archivo.

```
<form>
  <label for="files">Select files:</label>
  <input type="file" id="files" name="files" multiple>
</form>
```

Esto nos será muy útil sobre todo cuando permitamos subir archivos, ya que puede darse el caso que el usuario necesite subir 30 archivos a nuestro formulario y, sin este atributo, no podría.

Selección de ficheros: Qué hemos aprendido

1. Qué son, para qué sirven y cómo se usan los inputs de tipo file.
2. Los atributos que acompañan a este input.

Organización de campos

Estas serán las etiquetas que servirán para organizar los elementos del formulario, agrupándolos en categorías o temáticas, ordenar la información o para visualizar los elementos de forma más sencilla.

Organización de campos: Etiqueta fieldset

La etiqueta **<fieldset>** define una agrupación de elementos comunes dentro del formulario.

Siempre deberá ir acompañada de una etiqueta **<legend>** a la que se le asignará un título descriptivo. Este título es obligatorio por motivos de accesibilidad, pero puede ser ocultado mediante css.

```
<form action="/next-page" method="post">
  <fieldset>
    <legend>Datos Básicos</legend>
  </fieldset>
</form>
```

Organización de campos: Etiqueta label

Esta etiqueta siempre acompaña a un campo del formulario con un título descriptivo. Y debemos especificar con el atributo **for**. Y en el input usar el atributo **id** para relacionarlo.

El for del <label> y el id del <input> deberán coincidir, ya que de esta forma los relacionaremos y sería como decirle al formulario: "*ey! cuando encuentres el id "name" en alguno de los <input>, asígnale el <label> cuyo atributo for coincida*"

Vamos a verlo con un ejemplo.

```
<label for="name">Nombre</label>
<input type="text" name="name" id="name">
```

Organización de campos: Shortcuts

Para movernos entre inputs podemos usar la tecla TAB, que en este caso avanza hacia adelante los inputs y si pulsamos SHIFT + TAB volvemos hacia atrás. Esto nos será muy útil a la hora de rellenar formularios.

Organización de campos: Qué hemos aprendido

1. Qué etiquetas existen para ayudarnos a ordenar y agrupar la información de nuestros formularios.
2. Shortcuts que nos ayudarán a la hora de movernos por los formularios.

Botones de envío de formularios

Este tipo de input podemos decir que es de los más importantes, ya que si un formulario no lo contiene, el usuario no podrá enviar la información que ha rellenado, a no ser que pulse la tecla ENTER.

SUBMIT:Para crear un botón que nos envíe los datos del formulario al servidor tenemos el tipo **submit**. Su sintaxis es la siguiente:

```
<input type="submit" value="TextoBotón"/>
```

Una vez que pulsemos sobre el botón se enviarán los datos que el usuario haya insertado en el formulario siempre y cuando esté correctamente relacionado el formulario con nuestro backend a través del atributo action que ya vimos en la anterior sección.

Aquí también se puede usar una imagen como botón que enviará la información del formulario, si le añadimos un input de type image:

```
<form name="formulario" method="post" action="/send.php">
  <!-- Datos del formulario -->
  Usuario: <input type="text" name="usuario" />
  <!-- Botón de envío de formulario con imagen -->
  <input type="image" src="enviar.png" alt="Enviar" width="80" height="28" />
</form>
```

RESET: El otro tipo de botón con funcionalidad es el que nos permite el borrado de los datos del formulario. Para ello tenemos el tipo **reset**. La sintaxis de este botón será:

```
<input type="reset" value="TextoBotón"/>
```

Cuando el usuario pulse sobre el botón de borrado, todos los valores que el usuario haya insertado en el formulario se eliminarán.

Botones de envío de formularios: Botón personalizado

La etiqueta <button> nos será muy útil en javascript para desencadenar eventos de, por ejemplo, tipo clic.

RECORDAD que, cuando lo usemos para el envío de información, tenemos que añadirle el atributo **type=”submit”**

```
<button type="submit">Enviar</button>
```

Como hemos visto hasta ahora, los botones que hemos insertado han sido mediante el elemento input, si bien contamos con otro elemento para poner botones en el formulario que es el elemento <button>. Cuya funcionalidad es la misma que la del elemento input. La sintaxis del elemento <button> es:

```
<button name="nombre" type="TipoBoton" value="ValorBoton">Botón</button>
```

Dependiendo del tipo que asignamos al atributo type obtendremos un comportamiento u otro:

- **submit**, crea un botón para el envío de formulario.
- **reset**, crea un botón para el borrado de datos del formulario.
- **button**, crea un botón normal.

Botones de envío de formularios: Qué hemos aprendido

1. Cómo enviar la información recogida de un formulario a través de un botón.
2. Los types que puede recibir con cada una de sus funciones, como reset o submit.
3. La etiqueta <button> que será muy útil a la hora de interactuar con los eventos de javascript

Medidores y barras de progreso

Existen algunos otros controles que, aunque no nos permiten introducir información, pueden ser muy útiles en formularios para presentar información de una forma más visual u ofrecer datos adicionales al usuario. Eso sí, necesitaremos de javascript para crear elementos más complejos, con solo HTML serán muy sencillos.

Tenemos las barras de progreso con la etiqueta <progress>, que muestra por defecto una barra infinita que se mueve de izquierda a derecha. Esta será muy útil cuando queramos mostrarle al usuario el tiempo que queda para que se complete cierta tarea.

```
<form name="formulario" method="post" action="/send.php">
  <!-- Barra de progreso -->
  <progress max="100" value="10">
</form>
```

Con max indicamos el valor total de la barra y con value el calor actual.

Con <meter> podemos crear medidores personalizados. Un buen ejemplo sería el medidor de una página web que te dice si tu contraseña es débil, normal o buena en cuanto a seguridad.

Los atributos que acompañan a esta etiqueta <meter> son:

Atributo	Valor	Descripción
min	valor mínimo	Valor mínimo que puede alcanzar el medidor.
max	valor máximo	Valor máximo que puede alcanzar

		el medidor.
value	valor actual	Cantidad actual del medidor.
low	umbral bajo	Indica el umbral donde se considera bajo-medio.
high	umbral alto	Indica el umbral donde se considera medio-alto.
optimum	valor óptimo	Indica el valor óptimo del medidor.

Un ejemplo en código:

```
<form name="formulario" method="post" action="/send.php">
  <!-- Medidor -->
  <meter min="0" max="100"
         low="25" high="75"
         optimum="100" value="75">
</form>
```

Medidores y barras de progreso: Qué hemos aprendido

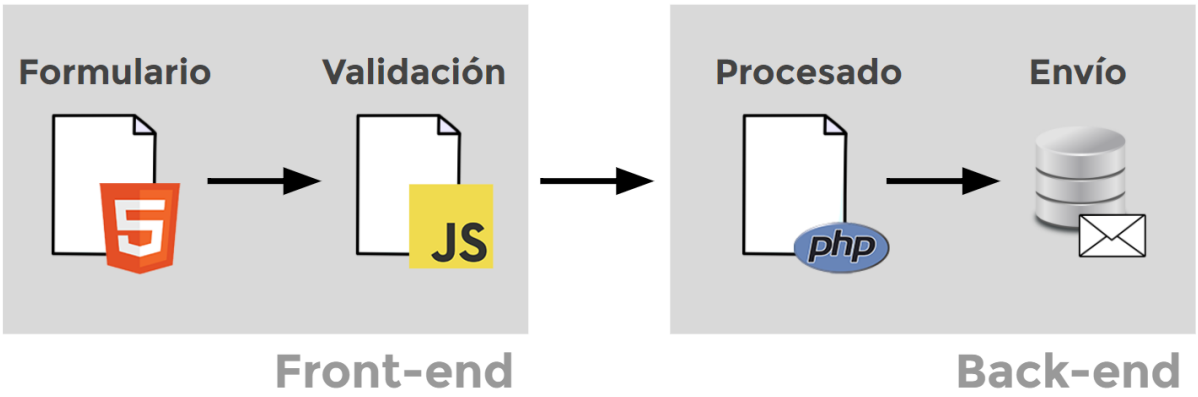
1. Cómo podemos crear barras de progreso y para qué son útiles.
2. Cómo crear medidores personalizados y sus atributos.

Validaciones HTML5

A la hora de crear una página web debemos tener en cuenta que el usuario puede equivocarse y es algo que ocurrirá, por lo tanto, tenemos herramientas para mitigar estos errores y que no afecten a la experiencia del usuario.

Estas herramientas son las llamadas herramientas de validación, que nos sirven para establecer pautas a la hora de indicar al usuario que la información que ha introducido es correcta o debe modificarla.

Este es un ejemplo típico de validación, en el que mediante javascript se comprueban los datos. Una vez validada se procesa y envía al backend.



Hay 4 tipos de validaciones:

1) Sin validación. El usuario podrá escribir la información que sea y no saltarán alarmas.

2) Validación en el front. Se verifica la información que introduce el usuario y se manda. Esto es peligroso ya que el usuario con conocimientos web podría saltarse esta validación.

3) Validación solo en el back. El usuario podrá enviar la información que sea pero el back la rechazará si no es válida. Esto no se recomienda ya que gasta muchos recursos y tiempo en los procesos innecesarios.

4) Doble validación en front y back. Es lo ideal.

Validaciones HTML5: Atributos más usados

Los más usados suelen ser los de medir la longitud de la cadena **minlength** o **maxlength** , campos requeridos **required** y valores mínimos y máximos con **min** o **max.**

required es la validación requerida indica que el campo es obligatorio en el formulario, y no podremos enviar los datos hasta rellenar dicho campo.

En el ejemplo podemos ver que es obligatorio que rellenemos nuestro nombre para poder enviar la información del formulario al backend.

```
<form method="post" action="/nombre">
  <label for="name">Nombre:</label>
  <input type="text" name="name" placeholder="Su nombre" required>
  <button type="submit">Enviar</button>
</form>
```

minlength y **maxlength** en los input de tipo **text** son usados para indicar un tamaño mínimo o máximo de caracteres en una entrada de datos. Así podremos restringir un input para que al menos tenga **n** caracteres, o que no se exceda de **n** caracteres.

En el ejemplo estamos diciendo que el input de tipo texto referente al “nick” debe ser mínimo de 2 caracteres de largo y máximo 8.

```
<form method="post" action="/user">
  <label for="nick">nickName:</label>
  <input type="text" name="nick" placeholder="Su nick" minlength="2" maxlength="8">
  <button type="submit">Enviar</button>
</form>
```

min y **max** para los **inputs** de type **number** que igual manera que con los campos de texto, podemos restringir los campos números mediante un valor máximo y un valor mínimo.

En este ejemplo, la edad que debemos rellenar podrá ir de los 14 a los 99 años.

```
<form method="post" action="/user">
    <label for="age">Edad:</label>
    <input type="number" name="age" placeholder="Su edad" min="14" max="99">
  <button type="submit">Enviar</button>
</form>
```

El atributo `disabled` deshabilita un campo del formulario, por lo que no se podrá hacer clic en él ni rellenar ningún tipo de dato. Además, sus valores nunca se enviarán junto al resto del formulario.

Y os preguntaréis para qué se usa entonces… pues son muy útiles a la hora de ir paso a paso en un formulario. Os habéis encontrado en muchas webs que hasta que no rellenas el campo nombre no os deja rellenar nada más… este tipo de lógica nos será muy útil para customizar los formularios.

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" disabled><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

El atributo `multiple` especifica que el usuario puede ingresar más de un valor en un campo. Tenéis que tener en cuenta que solo funciona con los siguientes tipos de entrada: correo electrónico y archivo.

```
<form>
  <label for="files">Select files:</label>
  <input type="file" id="files" name="files" multiple>
</form>
```

Esto nos será muy útil sobre todo cuando permitamos subir archivos, ya que puede darse el caso que el usuario necesite subir 30 archivos a nuestro formulario y, sin este atributo, no podría.

El atributo `readonly` indica que un campo de formulario es de solo lectura. Por lo tanto, no se podrá modificar.

CUIDADO! el valor que se introduzca en campos con este tipo de atributo se enviará junto al resto de información del formulario. Tened esto en cuenta a la hora de crear vuestras páginas.

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="Alberto" readonly><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Trevijano">
</form>
```

Validaciones HTML5: Customizar nuestro formulario

Si enlazas tu fichero con un archivo de estilos podemos customizar los inputs cuando son válidos, vamos a verlo porque siempre os gustan estas cositas.

Este ejemplo de código hará que si la información que introduce el usuario en nuestro input es válida, el color del fondo sea lightgreen, y, en caso de que sea inválido, el color de fondo será coral.

```
input:valid {
  background: lightgreen;
}

input:invalid {
  background: coral;
}
```

También se pueden dar estilos haciendo uso de los atributos de las etiquetas HTML, es decir.

Aquí, todos los inputs que tengan como tipo text tendrán un margen por debajo de 20px.

```
input[type="text"] {
  margin-bottom: 20px;
}
```

Dicho esto, NO es recomendable generalizar los estilos sobre los elementos.

```
.form-input:valid {
  background: lightgreen;
}

.form-input:invalid {
  background: red;
}

.form-input.form-input-text {
  margin-bottom: 20px;
}
```

Validaciones HTML5: Validación por Patrón

HTML5 nos permite usar **patterns**, vamos a ver algunos aspectos básicos de estos por si en algún momento os animáis a usarlos. Primero tenéis que saber que estos patrones se rigen por conjuntos de caracteres que cumplen con un patrón regex.

Las características básicas de expresiones regulares son:

Expresión	Carácter especial	Significado	Descripción
.	Punto	Comodín	Cualquier carácter (o texto de tamaño 1)
A|B	Pipe	Opciones lógicas	Opciones alternativas (o A o B)
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 carácter b puede aparecer o puede que no
\.	Barra invertida	Escape	El carácter . literalmente (no como comodín)

Os va parecer muy raro pero vamos a verlo con un ejemplo en un input con un patrón. Vamos a validar un nombre de usuario que solo permita letras y números.

```
<form method="post" action="/register">
  <label for="nombre">Usuario:</label>
  <input type="text" name="nombre"
      placeholder="Name" minlength="5" maxlength="40"
      required pattern="[A-Za-z0-9]+">
  <button type="submit">Registrarse</button>
</form>
```

Veamos un pequeño formulario con un pattern con un uso más práctico, imaginar que queremos que un usuario nos indique su modelo de Audi y los posibles son A1, A3, A4 y A6.

```
<form method="post" action="/coche">
    <label for="coche">Coche:</label>
    <input type="text" name="coche"
      placeholder="Su modelo de coche" required
      pattern="A|a(1|3|4|6)"
      title="Modelos posibles: A1, A3, A4 y A6">
  <button type="submit">Enviar</button>
</form>
```

En la documentación os dejamos una tabla para que intentéis construir vuestros propios pattern pero si os resulta complicado no hace falta que invirtáis mucho tiempo en estas validaciones.

EXPRESIÓN	CARÁCTER	DENOMINACIÓN	DESCRIPCIÓN
.	Punto	**Comodín**	Cualquier carácter
|	Pipe	**Opciones lógicas**	Alternativa (A | B)
()	Paréntesis	**Agrupaciones**	Agrupación C(A|B) =(CA o CB)
[*-*]	Corchetes	**Rango carácteres**	Dígito [0-9]=(0 al 9) [A-Z]=(A a la Z)
[^*-*]	^ en corchetes	**Rango exclusión**	[^A-Z]=(No letra A a Z mayúscula)
[*-*]*	Asterisco	**Cierre**	[1-9]*=(Un dígito repetido 0 ó más veces (vacío incluido))
[*-*]+	Signo más	**Cierre positivo**	[1-9]+=(Un dígito repetido 1 ó más veces)
[*-*]{*}	Llaves	**Coincidencia exacta**	[1-9]{3}=Cifra de 3 dígitos (dígito repetido 3 veces)
[*-*]{*,*}	Llaves	**Coincidencia de rango**	[1-9]{2,4}=Cifra de 2 a 4 dígitos (rep. de 2 a 4 veces)

***?**	Interrogación	**Carácter opcional**	b?=(El carácter b puede aparecer o puede que no)
*****	Barra invertida	**Escape**	\.=(El carácter . literalmente (no como comodín))

Si queréis probar algunas más, os dejamos una web donde podéis practicar: RegExR

Validaciones HTML5: Qué hemos aprendido

1. Qué son las validaciones HTML y para qué sirven.
2. Los tipos de validación que podemos encontrar.
3. Los atributos básicos que acompañan a las validaciones.
4. Los patrones que existen a la hora de hacer validaciones.