1

FORMULARIOS

Un formulario es un conjunto de controles que permiten al usuario interactuar con el documento o página. El objetivo de esta interacción suele ser, a menudo, la solicitud de información adicional o un mero intercambio de datos a petición del usuario.

Entre los diferentes controles que se pueden insertar o agregar a los formularios podemos encontrar acciones directas vinculadas a botones, solicitud de entradas de texto de una única línea, solicitud de entradas de texto multilínea, casillas de verificación, botones de única elección o tipo radio, selección de objetos y/o ficheros, entre otros.

En este capítulo vamos a ver los tipos básicos de formulario, cómo definirlos, cómo enviarlos, cómo hacerlos receptivos y, gran medida, cómo hacerlos usables y accesibles.

1.1 TIPOS DE FORMULARIO

Existen, fundamentalmente, cinco tipos de formulario que están definidos en función de su objetivo. Formularios de contacto, acceso, registro, suscripción y de entrada general.

No obstante, sea cual sea el tipo de formulario y su objetivo, para que cumpla con las expectativas de los clientes y con los requisitos establecidos por el RGPD (Reglamento General de Protección de Datos), deben incluir:

- Una casilla de verificación explícita con la que, los usuarios, puedan aceptar la política de privacidad del sitio. Por defecto, no puede estar seleccionada.
- Los enlaces pertinentes con la política de privacidad, el aviso legal, política de cookies, límite y responsabilidad, etcétera.
- Un texto, no demasiado largo, ni demasiado escueto, sobre qué datos se van a almacenar, quién será el responsable y cuál es el objetivo de dicho almacenamiento.
- Algún método para que el usuario ejerza su derecho a la eliminación de datos.
- Algún algoritmo de cifrado para que la información se gestione y manipule de forma encriptada. Esto es, que los datos que se envíen y/o almacenen estén codificados para evitar usos fraudulentos.

Únicamente se deben solicitar los campos que sean necesarios y estén bien justificados, es decir, no se debe solicitar una información concreta, como pueda ser tus hobbies, a no ser que se tenga una buena razón y esté justificada. Además, se debe tratar de conseguir que el medio sea lo suficientemente seguro y confiable, como para que invite a introducir los datos solicitados.

1.1.1 Formularios de contacto

Los formularios de contacto pueden llegar a ser un elemento clave en un sitio web porque permiten, o hacen posible, que la comunicación entre los usuarios y los proveedores sea directa y privada.

En general, un formulario de contacto debe solicitar sólo la información esencial para el proceso de comunicación. Esto es, no se deben pedir datos como la razón social, sitio web o teléfono sólo por fines comerciales o estadísticos.

Por ejemplo, el siguiente formulario podría ser considerado no fiable, además de ser ilegal.

Como se puede apreciar en la ilustración anterior, el sitio web no es un dato necesario y, aunque sea un dato opcional, puede suscitar desconfianza e impedir que el usuario haga uso de él. Además, como se ha mencionado anteriormente, es ilegal puesto que no presenta él checkbox de Política de Privacidad, entre otras cosas.

A continuación, se muestra el mismo ejemplo, pero corregido.

1.1.2 Formularios de suscripción

Los formularios de suscripción son, esencialmente, lo mismo que los formularios de contacto, pero más sencillos. Su objetivo es recuperar correos electrónicos para luego utilizarlos con fines comerciales o informativos.

En general, un formulario de suscripción sólo requiere de una caja de texto para recuperar el email, no obstante, es habitual pedir también el nombre para dirigirse a él. Por ejemplo, la siguiente ilustración podría una buena opción como formulario de suscripción.

1.1.3 Formularios de acceso

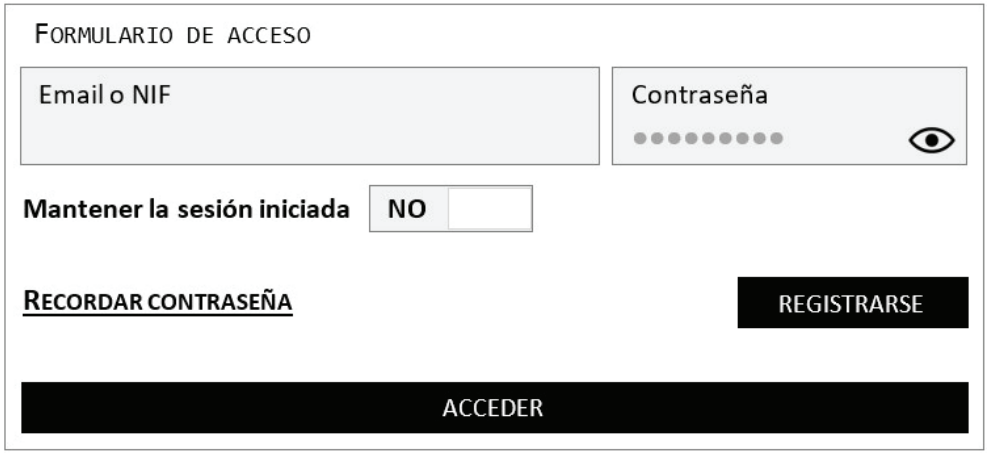
Los formularios de acceso son, junto con los formularios de contacto, los más recurrentes y utilizados en el mundo web. Su objetivo, como su propio nombre indica, es proporcionar acceso a información, productos o servicios que no están disponibles por vía pública o sin proporcionar una identidad.

En general, un formulario de acceso se caracteriza por tener dos cajas de texto para el nombre de usuario y contraseña, una casilla de verificación o botón de tipo interruptor para mantener la sesión iniciada, un enlace para recuperar la contraseña en supuesto caso de olvido y, evidentemente, un botón para acceder.

Los nombres de usuario se presentan con el texto visible y, a menudo, son el correo electrónico de los usuarios, aunque puede valer cualquier tipo de identificador único como un DNI, código de cliente, nombre de usuario, etcétera.

Las contraseñas suelen presentarse con el texto oculto, con asteriscos o puntos y, a menudo, suelen proporcionar un modo de cambiar a modo visible para ver el texto escrito.

A continuación, se muestra un ejemplo:



Si observamos la ilustración anterior, veremos que el campo contraseña tiene un icono de ojo que funciona como botón para cambiar el modo de presentación de la misma, Si pulsamos una vez se podrán en modo visible y seremos capaces de leer lo escrito, pero si pulsamos otra vez, volverá a ocultarse, mostrándose como la vemos ahora mismo.

Además, se ha puesto un botón de acceso al formulario de registro. Esta funcionalidad también es habitual, sobre todo, porque un registro también suele ser un acceso, una vez que ha finalizado el proceso.

1.1.4 Formularios de registro

Los formularios de registro suelen ir de la mano con los de acceso. Si bien, no siempre se ofrece la posibilidad de registrarse online, si es lo más habitual.

En general, un formulario de registro se caracteriza por la presencia de campos propietarios como el nombre o email del usuario, pero también pueden contener elementos considerados información sensible como son la edad, género o país de nacimiento. La decisión de qué campos se deben solicitar al usuario debe estar regida por la lógica de negocio y el RGPD (Reglamento General de Protección de Datos). No obstante, recordemos que, cuantos menos campos se solicite, más cómodo y confiable podrá sentirse el usuario.

Los nombres de usuario para dicho registro son, a menudo, el correo electrónico de los usuarios, aunque puede valer cualquier tipo de identificador único como un DNI o código de cliente.

Las contraseñas suelen seguir un patrón estricto de definición para evitar la fácil recuperación a través de métodos como la fuerza bruta y uso fraudulento.

En general, podríamos decir que un buen patrón de contraseña es aquel que tiene, al menos, un carácter en mayúsculas, un carácter en minúsculas, un dígito o número, un carácter especial como pueda ser el símbolo de almohadilla o interrogación y con una longitud mínima de ocho caracteres.

Sobra decir que no se deben utilizar sustituciones de carácter a número como pueda ser sustituir la letra A por el número cuatro, ni fechas especiales como cumpleaños o aniversarios y, por supuesto, nada de nombres propios. Eso sí, hay que tratar de que sea fácil de recordar y difícil de adivinar.

A continuación, se muestra un ejemplo:

Cabe destacar que, es bueno que los campos de un formulario de registro se soliciten de forma que el usuario vaya tomando confianza de menos a más. Es decir, se debe tratar de conseguir que los usuarios confíen un poco más cada vez que se avanza en el proceso de registro. Esto es así porque puede pasar que un usuario no se sienta del todo cómodo si se le solicita primero la edad antes que el nombre de usuario y contraseña.

Por último, sólo hay que destacar una cosa más. Si el formulario de registro requiere de muchos campos, es mejor que sean agrupados por contexto y solicitados en varios pasos o pantallas.

1.1.5 Formularios de entrada general

Los formularios de entrada general son aquellos formularios que tienen el propósito de recuperar una información concreta y que, por lo general, están pensados para cubrir las necesidades que no están cubiertas en las casuísticas anteriores.

Por ser algo más explícito, un formulario de propósito general podría ser un formulario de inserción de comentarios, de eventos o de inscripción a sorteos, concursos o juegos.

1.2 ELEMENTOS DISPONIBLES EN HTML5

1.2.1 Elemento form

El elemento `FORM` especifica que el contenido que se va a representar es un formulario, es decir, una estructura de interacción, formada por uno o varios elementos, que permiten introducir datos y enviarlos al servidor para su procesamiento.

Los formularios pueden albergar muy diversos elementos con diferentes formatos y estructuras, pero los más comunes quizás sean `INPUT`, `TEXTAREA`, `BUTTON`, `SELECT`, `OPTION`, `OPTGROUP`, `FIELDSET`, `LABEL` y `OUTPUT`, lo cuales se pasarán a ver a continuación.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
accept-charset	Especifica la codificación de caracteres que se debe usar cuando se realice el envío del formulario. Su valor predeterminado es `UNKNOWN`, que indica que la codificación de caracteres a utilizar debe ser la misma que la del documento actual. Entre sus valores más frecuentes podemos encontrar la codificación `ISO-8859-1` y la codificación `UTF-8`
action	Especifica el destino dónde enviar los datos, es decir, la dirección a la que se redirigirá cuando se vaya a realizar la acción de enviar los datos de formulario.
autocomplete	Especifica si los elementos de formulario deben ser manipulados y presentados por agente de usuario de manera automática o no. Sus valores son `ON` y `OFF`.
enctype	Especifica cómo se deben codificar los datos a enviar. Sus posibles valores son: • `APPLICATION/X-WWW-FORM-URLENCODED`: para indicar que se codifiquen todos los caracteres antes de enviarlos. Es la opción por defecto. • `MULTIPART/FORM-DATA`: para indicar que no se codifiquen los caracteres. • `TEXT-PLAIN`: para indicar que sólo los espacios sean codificados como símbolos de suma.

method	Especifica el método de envío por el que se deben enviar los datos. Los posibles valores son `GET` y con La diferencia entre uno y otro es que: • Mientras que el método `GET` envía los datos del formulario como parámetros de la propia dirección o URL, el método `POST` los agrega como parte del cuerpo de la solicitud HTTP. • Mientras que el método `GET` permite una longitud máxima de 3000 caracteres, el método `POST` no posee limitaciones de tamaño. • Mientras que el método `GET` permite la adición en la barra de direcciones o favoritos, el método `POST` no. **ⓘ NOTA** No se debe utilizar el método GET cuando se envían datos confidenciales o sensibles, sin embargo, puede ser la mejor opción cuando se trabaja con datos que no requieren de seguridad alguna, como es el caso de las QUERIES STRINGS o cadenas de consulta de Google.
name	Especifica o asigna el nombre del elemento. El atributo `NAME` puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice `CERO` el primero y, `N-1`, el último.
novalidate	Especifica si se deben validar los elementos del formulario o no. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, desactivará la validación de todos los elementos del formulario.
target	Específica a qué contexto o ventana se enviará la información. Entre los posibles valores que puede tomar, los más frecuentes son: • `_BLANK`: para indicar que se abra en una nueva pestaña o contexto, • `_SELF`: para indicar que se abra en la misma pestaña o contexto, • `_PARENT`: para indicar que se abra en la pestaña o contexto padre, • `_TOP`: para que se abra en el primer elemento `BODY` de la ventana. • `[NOMBRE]`: para que se abra en el marco identificado con ese nombre.

Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
    <label for="name">Nombre Completo:</label>
    <input type="text" id="name" name="name">

    <label>
        Nombre de usuario
        <input type="text" id="username" name="username">
    </label>

    <label for="password">Contraseña:</label>
    <input type="password" id="password" name="password">
```

```
    <label for="email">Email de contacto:</label>
    <input type="email" id="email" name="email">

    <label for="phone">Teléfono:</label>
    <input type="tel" id="phone" name="phone">

    <button type="submit">
        Guardar datos
    </button>
</form>
```

1.2.2 Elemento button

El elemento `BUTTON` especifica que el contenido que se va a representar es una acción. Como veremos, a diferencia del elemento `INPUT`, el elemento `BUTTON` puede albergar una gran variedad de contenidos como, por ejemplo, una imagen, un texto o, incluso, otros elementos HTML.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
autofocus	Especifica si se debe tomar el foco automáticamente después de ser renderizado y/o proceso de carga del documento. Sus valores pueden ser `TRUE` o `FALSE`.
disabled	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor `DISABLED.`
form	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo `ID` del elemento `FORM`, de lo contrario, no se hará efectivo.
formaction	Especifica el destino dónde enviar los datos, es decir, la dirección a la que se redirigirá cuando se vaya a realizar la acción de enviar los datos de formulario. También es importante destacar que, cuando este atributo está presente, el atributo `ACTION` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee que el formulario que se está definiendo puede ejecutar varias acciones que llevan a objetivos distintos.
formenctype	Especifica cómo se deben codificar los datos a enviar. También es importante destacar que, cuando este atributo está presente, el atributo `ENCTYPE` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario con diferentes tipos de codificación. Sus posibles valores son los mismos que los definidos en el atributo `ENCTYPE` del elemento `FORM`.

formmethod	Especifica el método de envío por el que se deben enviar los datos. También es importante destacar que, cuando este atributo está presente, el atributo `METHOD` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario con diferentes métodos. Sus posibles valores son los mismos que los definidos en el atributo `METHOD` del elemento `FORM`.
formnovalidate	Especifica si se deben validar los elementos del formulario o no. También es importante destacar que, cuando este atributo está presente, el atributo `NOVALIDATE` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee validar el formulario en función de si se ejecuta una u otra acción. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, desactivará la validación de todos los elementos del formulario.
formtarget	Específica a qué contexto o ventana se enviará la información. También es importante destacar que, cuando este atributo está presente, el atributo `TARGET` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario a diferentes ventanas o contextos. Sus posibles valores son los mismos que los definidos en el atributo `TARGET` del elemento `FORM`.
name	Especifica o asigna el nombre del elemento. El atributo `NAME` puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice `CERO` el primero y, `N-1`, el último.
type	Especifica el tipo de botón. Sus posibles valores son: • `SUBMIT`: para indicar que se envíen los datos del formulario, • `BUTTON`: para que permita hacer clic, pero no envíe los datos del formulario, • `RESET`: para indicar que se restablezcan todos los elementos del formulario a sus valores por defecto o preestablecidos.
value	Especifica el valor textual del elemento, por lo que no puede contener nada que no sea texto.

Ejemplo:

```
<button type="submit"
        formaction="./pages/login-test-2.php"
        formmethod="POST"
        formenctype="multipart/form-data"
        formtarget="_blank">
        value="Probar test 2"
</button>
```

1.2.3 Elemento datalist

El elemento `DATALIST` especifica que el contenido que se va a representar es una lista de opciones predefinidas para un elemento `INPUT`.

La razón para utilizar este elemento es para proveer de una funcionalidad de autocompletado a los elementos `INPUT`. Los usuarios podrán ver las diferentes opciones como si de un desplegable se tratase, pero con la opción de ir buscando a través de coincidencias parciales proporcionadas mediante teclado.

Entre los atributos que admite en su configuración, cabe destacar que, el único atributo que necesita para funcionar, es el atributo `ID`. El valor de este atributo debe coincidir exactamente con el valor del atributo `LIST` declarado en el elemento `INPUT` al que está asociado.

Ejemplo:

```
<input list="technologies">

<datalist id="technologies">
  <option value="HTML5">
  <option value="JavaScript">
  <option value="CSS3">
  <option value="SVG">
</datalist>
```

1.2.4 Elemento fieldset

El elemento `FIELDSET` especifica que el contenido que se va a representar es una agrupación de elementos relacionados. En general, y salvo excepciones, todos los agentes de usuario dibujan un cuadro alrededor de este elemento, equivalente a un estilo de borde.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
disabled	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor `DISABLED`.
form	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo `ID` del elemento `FORM`, de lo contrario, no se hará efectivo.

name	Especifica o asigna el nombre del elemento. El atributo `NAME` puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice `CERO` el primero y, `N-1`, el último.

Ejemplo:

```
<form action="./login.php">
    <fieldset>
        <legend>Acceso Privado</legend>

        <label for="username">Nombre de usuario</label>
        <input id="username" name="username" />
        <label for="password">Contraseña</label>
        <input id="password" name="password" />

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

1.2.5 Elemento input

El elemento `INPUT` especifica que el contenido que se va a representar es una entrada de datos.

Una de las cualidades más importantes que tiene el elemento `INPUT` es que tiene una gran variedad de validaciones nativas. Por ejemplo, es posible definir un elemento de entrada que sólo admita números, que sólo admita fechas en un formato específico o que valide las reglas de nombres de los correos electrónicos.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
accept	Especifica los tipos de archivo válidos cuando se utiliza un elemento `INPUT` de tipo `FILE.` Todos los posibles valores que puede tomar este atributo pueden encontrarse en IANA Media Types (*http://www.iana.org/assignments/media-types/*), aunque puede tomar un valor comodín como es `AUDIO/*, VIDEO/*` o `IMAGE/*`, que indican que se aceptan todas las extensiones de archivo de cada tipo.
alt	Permite especificar un texto alternativo para las imágenes que son definidas a través de un `INPUT` tipo `IMAGE.`

autocomplete	Especifica si los elementos de formulario deben ser manipulados y presentados por agente de usuario de manera automática o no. Sus valores son `ON` y `OFF`.
autofocus	Especifica si se debe tomar el foco automáticamente después de ser renderizado y/o proceso de carga del documento. Sus valores pueden ser `TRUE` o `FALSE`.
checked	Especifica si el elemento está chequeado o no. Cabe destacar que este atributo sólo es válido para los elementos `INPUT` de tipo `RADIO` y `CHECKBOX` y tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento se marcará como chequeado o seleccionado.
disabled	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor `DISABLED`.
form	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo `ID` del elemento `FORM`, de lo contrario, no se hará efectivo.
formaction	Especifica el destino dónde enviar los datos, es decir, la dirección a la que se redirigirá cuando se vaya a realizar la acción de enviar los datos de formulario. También es importante destacar que, cuando este atributo está presente, el atributo `ACTION` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee que el formulario que se está definiendo puede ejecutar varias acciones que llevan a objetivos distintos.
formenctype	Especifica cómo se deben codificar los datos a enviar. También es importante destacar que, cuando este atributo está presente, el atributo `ENCTYPE` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario con diferentes tipos de codificación. Sus posibles valores son los mismos que los definidos en el atributo `ENCTYPE` del elemento `FORM`.
formmethod	Especifica el método de envío por el que se deben enviar los datos. También es importante destacar que, cuando este atributo está presente, el atributo `METHOD` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario con diferentes métodos. Sus posibles valores son los mismos que los definidos en el atributo `METHOD` del elemento `FORM`.
formnovalidate	Especifica si se deben validar los elementos del formulario o no. También es importante destacar que, cuando este atributo está presente, el atributo `NOVALIDATE` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee validar el formulario en función de si se ejecuta una u otra acción. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, desactivará la validación de todos los elementos del formulario.

formtarget	Específica a qué contexto o ventana se enviará la información. También es importante destacar que, cuando este atributo está presente, el atributo `TARGET` del elemento `FORM` es anulado. No obstante, será una buena herramienta cuando se desee enviar un mismo formulario a diferentes ventanas o contextos. Sus posibles valores son los mismos que los definidos en el atributo `TARGET` del elemento `FORM`.
height	Especifica el alto del elemento en píxeles.
list	Especifica la lista de opciones predefinidas que se debe asociar a un elemento `INPUT`. Cabe destacar que, para que un elemento `INPUT` acepte las opciones predefinidas del elemento `DATALIST`, el valor del atributo `ID` del `DATALIST` debe ser el mismo que el valor del atributo `LIST` del elemento `INPUT`.
max	Especifica el límite superior del rango de valores aceptado, es decir, el valor máximo que puede aceptar o se puede introducir en el elemento. Aunque este atributo es válido para los elementos `INPUT` de tipo `METER` y `PROGRESS`, no todos los agentes de usuario lo soportan. Internet Explorer 10 es uno de ellos.
maxlength	Especifica la longitud máxima, en caracteres, del valor que puede ser ingresado en el elemento.
min	Especifica el límite inferior del rango de valores aceptado, es decir, el valor mínimo que puede aceptar o se puede introducir en el elemento. Aunque este atributo es válido para los elementos `INPUT` de tipo `METER` y `PROGRESS`, no todos los agentes de usuario lo soportan. Internet Explorer 10 es uno de ellos.
multiple	Especifica que el elemento admite la selección o inserción de más de un valor. Cabe destacar que este atributo sólo es válido para los elementos `INPUT` de tipo `FILE`. Además, tendrá efecto con sólo declararlo, es decir, en cuanto esté presente e independientemente de su valor, el elemento permitirá la selección de múltiples valores.
name	Especifica o asigna el nombre del elemento. El atributo `NAME` puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice `CERO` el primero y, N-1, el último.
pattern	Especifica la expresión regular con la que se validará la entrada de datos en el elemento. La forma de trabajar con expresiones regulares es similar a la de JavaScript, no obstante, en la web de HTML5 Pattern, ubicada en la dirección *http://html5pattern.com/,* se pueden encontrar multitud de ejemplos aptos para ser utilizados. Cabe destacar que este atributo sólo es válido para los elementos `INPUT` de tipo `TEXT, DATE, SEARCH, URL, TEL, EMAIL` y `PASSWORD`.

placeholder	Especifica el texto que se debe mostrar cómo pista de datos válidos cuando el elemento no contenga un valor. Aunque algunos agentes de usuario pueden no aceptar este atributo, en general es una buena idea hacerlo para ayudar a la usabilidad y accesibilidad web. Si va asociado con otro atributo como `PATTERN`, la pista a proporcionar debe estar en concordancia con la validez de la entrada.
readonly	Especifica que el elemento es de sólo lectura, es decir, que no permite la inserción de nuevos valores ni la modificación de su valor actual. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, provocará que se ponga en modo de sólo lectura. No obstante, no quiere decir que no se pueda seleccionar, resaltar o copiar.
required	Especifica que el elemento debe contener valor, aunque este sea un espacio en blanco, es decir, que no puede ser nulo ni vacío.
size	Especifica la ubicación del recurso externo que se desea cargar o mostrar. Cabe destacar que este atributo sólo es válido para los elementos `INPUT` de tipo `IMAGE`.
src	Especifica la ubicación del recurso externo que se desea cargar o mostrar. Este atributo sólo es válido para los elementos `INPUT` de tipo `IMAGE`.
step	Especifica el intervalo de incremento o decremento sobre el valor actual del elemento, es decir, el número de pasos que se deben sumar o restar al valor actual del elemento. Cabe destacar que este atributo sólo es válido para los elementos `INPUT` de tipo `NUMBER, RANGE, DATE, DATETIME, DATTETIME-LOCAL, MONTH, TIME`, y `WEEK`.
type	Especifica el tipo de elemento `INPUT`. Sus posibles valores son: • `BUTTON`: Para indicar que es un botón. • `CHECKBOX`: Para indicar que es una casilla de verificación. • `DATE`: Para indicar que es una fecha. • `DATETIME`: Para indicar que es una fecha con hora. Sólo está soportado por Safari y Opera. • `DATETIME-LOCAL`: Para indicar que es una fecha con hora sin zona horaria. Sólo está soportado por Chrome, Microsoft Edge y Opera. • `EMAIL`: Para indicar que es un correo electrónico. • `FILE`: Para indicar que es un control para subir archivos al servidor. • `HIDDEN`: Para indicar que es un campo oculto. • `IMAGE`: Para indicar que es una imagen. • `MONTH`: Para indicar que es un mes. Sólo está soportado por Chrome, Microsoft Edge y Opera. • `NUMBER`: Para indicar que es un número real. • `PASSWORD`: Para indicar que es una contraseña. • `RADIO`: Para indicar que es un botón de radio. • `RESET`: Para indicar que es un botón de reinicialización que establece los valores por defecto o preestablecidos. • `SEARCH`: Para indicar que es un campo de búsqueda. • `TEL`: Para indicar que es un teléfono Actualmente sólo soportado por Safari. • `TEXT`: Para indicar que es una caja o campo de texto. • `TIME`: Para indicar que es una hora. Actualmente no soportado por Safari. • `URL`: Para indicar que es una dirección web o URL. • `WEEK`: Para indicar que es un valor de semana. Sólo está soportado por Chrome, Microsoft Edge y Opera.

value	Especifica el valor textual del elemento, por lo que no puede contener nada que no sea texto.
width	Especifica el ancho del elemento en píxeles.

Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
    <label for="field1">Campo de texto:</label>
    <input type="text" id="field1" name="field1">

    <label for="field2">Campo sólo números:</label>
    <input type="number" id="field2" name="field2">

    <label for="field3">Campo fecha:</label>
    <input type="date" id="field3" name="field3">

    <label for="field4">Campo URL:</label>
    <input type="url" id="field4" name="field4">

    <label for="field5">Campo para emails:</label>
    <input type="email" id="field5" name="field5">

    <label for="field6">Campo para teléfonos:</label>
    <input type="tel" id="field6" name="field6">

    <label for="field7">Campo para imágenes:</label>
    <input type="image" id="field7" name="field7">

    <label for="field8">Elemento para adjuntar archivos:</label>
    <input type="file" id="field8" name="field8">

    <label for="field9">Casilla de verificación:</label>
    <input type="checkbox" id="field9" name="field9">

    <label for="field10">Campo tipo radio:</label>
    <input type="radio" id="field10" name="field10">

    <label for="field11">Campo oculto:</label>
    <input type="hidden" id="field11" name="field11">

    <label for="submit">Botón enviar:</label>
    <input type="submit" id="submit" name="submit">

    <button type="submit">
        Guardar datos
    </button>
</form>
```

1.2.6 Elemento label

El elemento `LABEL` especifica que el contenido que se va a representar es una etiqueta para un elemento de formulario `INPUT`, `METER`, `PROGRESS`, `SELECT` o `TEXTAREA`.

Cuando se utiliza elemento `LABEL`, la estructura del documento se vuelve más legible y consistente. Además, beneficia tanto a la usabilidad web, como a la accesibilidad web porque los elementos pequeños como son las casillas de verificación pueden manipularse a través del elemento `LABEL`.

Además, al tener el control de formulario una etiqueta asociada, las herramientas de asistencia, como los lectores de pantalla, pueden leerla y comunicárselo a los usuarios con discapacidad visual total o parcial.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
for	Especifica el `ID` del elemento de formulario con el que está vinculada la etiqueta. El contenido al que se hace referencia a través de este atributo funcionará como texto descriptivo de la entrada de datos.
form	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo `ID` del elemento `FORM`, de lo contrario, no se hará efectivo.

Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
    <label for="name">Nombre:</label>
    <input type="text" id="name" name="name">

    <!-- ... -->
</form>
```

1.2.7 Elemento legend

El elemento `LEGEND` especifica que el contenido que se va a representar es un título para un elemento `FIELDSET`.

Ejemplo:

```
<form action="./login.php">
    <fieldset>
        <legend>Acceso a ejemplo.com</legend>
        <!-- ... -->
    </fieldset>
</form>
```

Aunque el uso del elemento `LEGEND` no está recomendado fuera del elemento `FIELDSET`, se puede aplicar siempre que se desee. No obstante, a efectos de semántica web, sólo tendrá efecto cuando se encuentre dentro de un elemento `FIELDSET`.

1.2.8 Elemento meter

El elemento `METER` especifica que el contenido que se va a representar es una medición escalar, es decir, como una barra de progreso con rango conocido o valor fraccional.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
form	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo `ID` del elemento `FORM`, de lo contrario, no se hará efectivo.
high	Especifica el valor a partir del cual se considera que es un valor alto, siendo este, igual o menor que el valor del atributo `MAX`.
low	Especifica el valor a partir del cual se considera que es un valor bajo, siendo este, igual o menor que el valor del atributo `MIN`.
max	Especifica el límite superior del rango de valores aceptado, es decir, el valor máximo que puede aceptar o se puede introducir en el elemento.
min	Especifica el límite inferior del rango de valores aceptado, es decir, el valor mínimo que puede aceptar o se puede introducir en el elemento.
optimum	Especifica el valor a partir del cual se considera que es un valor optimo, siendo este, mayor que el valor del atributo `MIN` y menor que el atributo `MAX`.
value	Especifica el valor numérico inicial del elemento.

Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
    <label for="available-space">Espacio disponible en disco:</label>
    <meter id="available-space" value="324" min="0" max="1024">
        324MB libres de 1024 MB
    </meter>

    <!-- ... -->
</form>
```

Cabe destacar que, aunque pueden parecerse, una medición escalar no es una barra de progreso. Por esta razón, el elemento `METER` no debe utilizarse para mostrar valores de progreso. El correcto uso de este elemento es para uso o capacidad de disco, para indicar el valor de tareas que tuvieron éxito en un conjunto definido o situaciones similares.

1.2.9 Elemento optgroup

El elemento `OPTGROUP` especifica que el contenido que se va a representar es un grupo de opciones relacionadas de un elemento `SELECT`.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
disabled	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor `DISABLED`.
label	Especifica la etiqueta que se mostrará para diferenciar el grupo de opciones relacionadas. Este atributo sólo es aplicable para el elemento `OPTGROUP`, descendiente directo del elemento `SELECT`.

Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
    <label for="motorcycles">Motos:</label>
    <select id="motorcycles">
        <optgroup label="Harley Davidson">
            <option value="01">Sportster</option>
            <option value="02">Softail</option>
            <option value="03">Touring</option>
        </optgroup>
        <optgroup label="Indian">
            <option value="04">Chief</option>
            <option value="05">Springfield</option>
            <option value="06">Scout Sixty</option>
        </optgroup>
    </select>

    <!-- ... -->
</form>
```

1.2.10 Elemento option

El elemento `OPTION` especifica que el contenido que se va a representar es una opción para elemento `SELECT`.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
disabled	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor `DISABLED`.
label	Especifica la etiqueta que se mostrará para identificar o etiquetar la opción. Cuando este atributo está presente, su valor será mostrado en vez de su contenido interno. No obstante, no todos los agentes de usuario proporcionan soporte nativo a este atributo.
selected	Especifica si la opción debe marcarse como seleccionada.
value	Especifica el valor textual del elemento, por lo que no puede contener nada que no sea texto.

Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
    <label for="rate">Calificación:</label>
    <select id="rate">
        <option value="01">Mala</option>
        <option value="02">Media</option>
        <option value="03">Buena</option>
    </select>

    <!-- ... -->
</form>
```

1.2.11 Elemento output

El elemento `OUTPUT` especifica que el contenido que se va a representar es el resultado de una operación. Esto puede ser una buena opción a implementar cuando se trata de mostrar el resultado de una operación en dónde el usuario introduce varios valores a través de elementos `INPUT`.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
for	Especifica los `ID` de los elementos de formulario con los que se operará para mostrar el resultado en el elemento `OUTPUT`.
form	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo `ID` del elemento `FORM`, de lo contrario, no se hará efectivo.

name	Especifica o asigna el nombre del elemento. El atributo NAME puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice CERO el primero y, N-1, el último.

Ejemplo:

```
<form oninput="res.value=parseInt(op1.value) + parseInt(op2.value)">0
    <label>Suma de dos operandos</label>

    <input type="range" id="op1" value="50">100
    +
    <input type="range" id="op2" value="50">
    =
    <output name="res" for="op1 op2">100</output>
</form>
```

1.2.12 Elemento progress

El elemento PROGRESS especifica que el contenido que se va a representar es una barra de progreso.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
max	Especifica el límite superior del rango de valores aceptado, es decir, el valor máximo que puede aceptar o se puede introducir en el elemento.
value	Especifica el valor numérico inicial del elemento.

Ejemplo:

```
<form oninput="res.value=parseInt(op1.value) + parseInt(op2.value)">0
    <label>Progreso de instalación</label>
    <progress id="progress" value="54" max="100"> 54% </progress>
</form>
```

Cabe destacar que, aunque puedan parecerse, una barra de progreso no es una medición escalar. Por esta razón, el elemento PROGRESS no debe utilizarse para mostrar valores indicadores. El correcto uso de este elemento es únicamente para mostrar cómo progresa, o ha progresado, una tarea o proceso.

1.2.13 Elemento select

El elemento `SELECT` especifica que el contenido que se va a representar es un desplegable con una lista de opciones predefinida.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
autofocus	Especifica si se debe tomar el foco automáticamente después de ser renderizado y/o proceso de carga del documento. Sus valores pueden ser `TRUE` o `FALSE`.
disabled	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor `DISABLED`.
form	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo `ID` del elemento `FORM`, de lo contrario, no se hará efectivo.
multiple	Especifica que el elemento admite la selección o inserción de más de un valor. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente e independientemente de su valor, el elemento permitirá la selección de múltiples valores.
name	Especifica o asigna el nombre del elemento. El atributo `NAME` puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice `CERO` el primero y, `N-1`, el último.
required	Especifica que el elemento debe contener valor, aunque este sea un espacio en blanco, es decir, que no puede ser nulo ni vacío.
size	Especifica el ancho en caracteres que debe tener el elemento.

Ejemplo:

```
<form action="/action_page.php" method="get" name="frm">
    <label for="rate">Calificación:</label>
    <select id="rate">
        <option value="01">Mala</option>
        <option value="02">Media</option>
        <option value="03">Buena</option>
    </select>

    <!-- ... -->
</form>
```

1.2.14 Elemento textarea

El elemento `TEXTAREA` especifica que el contenido que se va a representar es una caja de texto con la opción de multilínea, es decir, un control de entrada de datos de múltiples líneas.

Entre los atributos que admite en su configuración, se deben destacar los siguientes:

Atributo	*Descripción*
autofocus	Especifica si se debe tomar el foco automáticamente después de ser renderizado y/o proceso de carga del documento. Sus valores pueden ser `TRUE` o `FALSE`.
cols	Especifica la anchura, en caracteres, del elemento.
disabled	Especifica si el elemento está deshabilitado o no. Esta propiedad tendrá efecto con sólo declararla, es decir, en cuanto esté presente e independientemente de su valor, el elemento aparecerá deshabilitado. No obstante, suele declararse con el valor `DISABLED`.
form	Especifica el formulario al que pertenece el elemento cuando se declaran sus elementos fuera del ámbito del formulario. Con respecto a sus posibles valores, el valor de este atributo debe ser el mismo que el indicado por el atributo `ID` del elemento `FORM`, de lo contrario, no se hará efectivo.
maxlength	Especifica la longitud máxima, en caracteres, del valor que puede ser ingresado en el elemento.
name	Especifica o asigna el nombre del elemento. El atributo `NAME` puede ser utilizado para recuperar el valor de los campos del formulario desde el lado del servidor y puede resultar muy útil cuando se desean manipular los elementos desde JavaScript o CSS. Si el valor de este atributo lleva asignado como sufijo [] (los símbolos de corchetes), el identificador de nombre se definirá y actuará como si de un array numérico se tratase, es decir, se podrán referenciar los distintos elementos a través de un índice, siendo, el índice `CERO` el primero y, `N-1`, el último.
placeholder	Especifica el texto que se debe mostrar cómo pista de datos válidos cuando el elemento no contenga un valor. Aunque algunos agentes de usuario pueden no aceptar este atributo, en general es una buena idea hacerlo para ayudar a la usabilidad y accesibilidad web. Si va asociado con otro atributo como `PATTERN`, la pista a proporcionar debe estar en concordancia con la validez de la entrada.
readonly	Especifica que el elemento es de sólo lectura, es decir, que no permite la inserción de nuevos valores ni la modificación de su valor actual. Cabe destacar que este atributo tendrá efecto con sólo declararlo, es decir, en cuanto esté presente, e independientemente de su valor, provocará que se ponga en modo de sólo lectura. No obstante, no quiere decir que no se pueda seleccionar, resaltar o copiar.
required	Especifica que el elemento debe contener valor, aunque este sea un espacio en blanco, es decir, que no puede ser nulo ni vacío.
rows	Especifica la altura del elemento en líneas de caracteres.

wrap	Especifica si el texto debe incluir o no la definición de nuevas líneas cuando se realiza el envío del formulario para que se ajuste al ancho del elemento establecido. Sus posibles valores son `HARD` y `SOFT`, los cuales indican que se agreguen nuevas líneas al texto enviado o no, respectivamente. El carácter de nueva línea añadido al texto será en base al valor del atributo `COLS`, es decir, cuando el texto llegue a la longitud establecida por `COLS`, agregará o no un nuevo salto de línea en función de si el atributo `WRAP` está o no establecido a `HARD`. El uso de este atributo no se recomienda si se desea mantener un buen nivel de usabilidad y accesibilidad web puesto que puede provocar pérdidas en la legibilidad de los datos si no se utiliza adecuadamente.

Ejemplo:

```
<form action="/action_page.php" method="post" name="frm">
    <label>
        Descripción
        <textarea id="desc" rows="24" cols="80"></textarea>
    </label>

    <!-- ... -->
</form>
```

1.3 ELEMENTOS DISPONIBLES EN CSS

CSS no presenta ninguna propiedad específica para elementos de formulario, sin embargo, cabe destacar que estos elementos pueden aprovecharse de las mismas ventajas que cualquier otro elemento de bloque o caja. Es decir, pueden ser personalizados con propiedades de todo tipo, como son `FONT-FAMILY`, `TEXT-ALIGN`, `POSITION`, `DISPLAY`, `WIDTH`, `HEIGHT`, `PADDING`, `MARGIN`, `BORDER`, `BACKGROUND`, ...

No obstante, entre todas ellas, cabe destacar dos:

1.3.1 Propiedad box-sizing

Especifica cómo deben asignarse y calcularse el alto y ancho de los elementos. Esto es, si deben incluir los márgenes internos (padding) y/o los bordes, o no. Entre sus posibles valores podemos encontrar:

- **CONTENT-BOX**: Indica que se debe incluir sólo el contenido, e ignorar los márgenes internos y bordes. Es el valor por defecto.
- **BORDER-BOX**: Indica que se deben incluir el contenido, padding y bordes.

ⓘ NOTA

En general, se puede afirmar que, trabajar con cajas o capas incluyendo los márgenes internos y los bordes es más fácil de manejar y facilita el diseño adaptativo, aunque no siempre.

1.3.2 Propiedad resize

Especifica si los elemento `TEXTAREA` deben permitir la manipulación del tamaño del elemento desde la interfaz que presenta el agente de usuario. Sus posibles valores son:

- **NONE**: Indica que el usuario no puede cambiar el tamaño del elemento.
- **HORIZONTAL**: Indica que el usuario puede cambiar, únicamente, el tamaño del elemento horizontalmente.
- **VERTICAL**: Indica que el usuario puede cambiar, únicamente, el tamaño del elemento verticalmente.
- **BOTH**: Indica que el usuario puede cambiar el tamaño del elemento en ambas direcciones.

ⓘ NOTA

Aunque esta propiedad puede resultar muy útil, su soporte está muy limitado. De hecho, no está soportado por Internet Explorer, Microsoft Edge 78 o inferior, ni Opera 14 o inferior.

Ejemplo:

```
textarea { resize: none; }
```

1.4 VALIDACIÓN DE FORMULARIOS

La validación de formularios se realiza a través de JavaScript y, hasta no hace tanto, no era casi personalizable ni eficiente. Ahora, sin embargo, gracias a la amplia gama de propiedades que poseen los elementos de formulario, junto con los métodos de notificación y manipulación que nos provee HTML5, podemos realizar validaciones de forma bastante rápida y sencilla.

1.4.1 La interfaz validitystate

La interfaz `VALIDITYSTATE` es un objeto que representa todos los posibles esta-dos por los que puede pasar un elemento de formulario. Además, suele indicar la razón o el motivo por el que se encuentra en ese estado.

Entre sus propiedades más frecuentes podemos encontrar:

Propiedad	*Descripción*
badInput	Esta propiedad devuelve true si ha habido algún problema con la conversión del dato introducido.
customError	Esta propiedad devuelve `TRUE` si el elemento contiene asignado un error definido por el usuario establecido a través del método `SETCUSTOMVALIDITY`.
patternMismatch	Esta propiedad devuelve true si el elemento no cumple el patrón definido. Si su valor es true, la pseudo-clase `:INVALID` de CSS también se activará.
rangeOverflow	Esta propiedad devuelve true si el elemento contiene un valor mayor al provisto por la propiedad `MAX`. Si su valor es `TRUE`, las pseudo-clases `:INVALID` y `:OUT-OF-RANGE` de CSS también se activarán.
rangeUnderflow	Esta propiedad devuelve true si el elemento contiene un valor menor al provisto por la propiedad `MIN`. Si su valor es `TRUE`, las pseudo-clases `:INVALID` y `:OUT-OF-RANGE` de CSS también se activarán.
stepMismatch	Esta propiedad devuelve `TRUE` si el elemento contiene un valor que no concuerda con el paso provisto por la propiedad `STEP`. Si su valor es `TRUE`, las pseudo-clases `:INVALID` y `:OUT-OF-RANGE` de CSS también se activarán.
tooLong	Esta propiedad devuelve `TRUE` si el elemento tiene una longitud mayor que la provista por el atributo `MAXLENGTH`. Si su valor es `TRUE`, las pseudo-clases `:INVALID` y `:OUT-OF-RANGE` de CSS también se activarán.
tooShort	Esta propiedad devuelve `TRUE` si el elemento tiene una longitud menor a la provista por el atributo `MINLENGTH`. Si su valor es `TRUE`, las pseudo-clases `:INVALID` y `:OUT-OF-RANGE` de CSS también se activarán.
typeMismatch	Esta propiedad devuelve `TRUE` si el elemento contiene una sintaxis incorrecta. Sólo es válido para los tipos de `INPUT EMAIL` y `URL`. Si su valor es `TRUE`, la pseudo-clase `:INVALID` de CSS también se activará.
valid	Esta propiedad devuelve `TRUE` si el elemento cumple todas las restricciones requeridas. Si su valor es `TRUE`, la pseudo-clase `:VALID` de CSS también se activará.
valueMissing	Esta propiedad devuelve `TRUE` si el elemento es requerido y se encuentra vacío. Si su valor es `TRUE`, la pseudo-clase `:INVALID` de CSS también se activará.

1.4.2 Propiedades y métodos

A continuación, se muestran los principales métodos y propiedades que pueden ser utilizados en la validación de formularios.

1.4.2.1 PROPIEDAD VALIDITY

Esta propiedad resulta ser un objeto que devuelve un conjunto de datos de tipo `VALIDITYSTATE` y permite conocer el resultado de todos los posibles problemas que se han producido tras realizar una comprobación de validación.

1.4.2.2 MÉTODO SETCUSTOMVALIDITY

El método `SETCUSTOMVALIDITY` permite definir mensajes de error personalizados en los elementos de formulario. El mensaje, proporcionado como parámetro, es guardado en la propiedad `VALIDATIONMESSAGE`.

1.4.2.3 PROPIEDAD VALIDATIONMESSAGE

Esta propiedad contiene el mensaje generado tras el proceso de validación. Si el elemento no ha sido validado aún o ha pasado con éxito todo el proceso de validación, el contenido de esta propiedad estará establecida a cadena vacía. Si, por el contrario, existe algún error o problema con la validación, esta propiedad mostrará el mensaje de error o información sobre el problema.

1.4.2.4 MÉTODO CHECKVALIDITY

Este método comprueba si se cumplen las restricciones que tiene definido el elemento de formulario. Si todo es correcto, es decir, que pasa la validación, devolverá `TRUE`, en cualquier otro caso, devolverá `FALSE`.

1.4.3 Eventos

A continuación, se muestran los principales eventos que pueden ser utilizados en la validación de formularios.

1.4.3.1 EVENTO INVALID

Cada vez que se solicita la acción de enviar un formulario, se realiza una acción de validación previamente. Si el proceso de validación no tuvo éxito, el navegador lanza una especie de excepción que marca al elemento como inválido y le asigna la pseudo-clase de CSS `:INVALID`.

Pues bien, si además de controlar la validación interna queremos o necesitamos gestionarla de forma externa, para esto, tenemos el evento `INVALID`.

El evento `INVALID` es lanzado cuando el elemento realiza el proceso de validación y no cumple alguna de sus restricciones. Una vez, dentro de este evento podemos, por ejemplo, mostrar los mensajes personalizados que hemos crea-do para nuestra aplicación en el lugar de los predefinidos.

1.4.3.2 EJEMPLO DE VALIDACIÓN

Imaginemos que deseamos comprobar que el valor introducido en un `INPUT` denominado `STATUS` concuerde con uno de los tres posibles valores: "Asignado", "En Progreso" o "Finalizado".

Para ello, lo primero que necesitaremos es declarar el código HTML con un `LABEL`, un campo de entrada de texto y dos elementos adicionales para poder establecer los posibles mensajes de error.

```
<h1>Prueba de validación</h2>
<form name="frm" enctype="multipart/form-data" method="get">
    <label>
       Estado:
       <input id="status" type="text" oninput="check(this)" />
    </label>

    <h2>Mensaje tras validación</h2>
    <div id="validateMsg"></div>

    <h2>Listado de errores de validity</h2>
    <div id="validity"></div>
</form>
```

Una vez insertado el HTML, necesitamos declarar la funcionalidad de JavaScript que realizará la validación. Para ello, podríamos hacer algo como:

```
function check(input) {
    // Recuperamos el valor introducido y los elementos a manipular
    var val = input.value;
    var vm  = document.getElementById("validateMsg");
    var va  = document.getElementById("validity");

    // Comprobamos si el valor es válido
    if (input.value != "Asignado" &&
        input.value != "En Progreso" &&
        input.value != "Finalizado") {

        // Como no es válido, establecemos un mensaje de error entendible
        // y se lo asignamos a la función setCustomValidity
        var msg = '"' + val + '" no es un estado.';
        input.setCustomValidity(msg);
        // Asignamos el mensaje de error a nuestro objeto validityMsg
        vm.innerHTML = input.validationMessage;
        va.innerHTML = '';
```

```
        // Mostramos el listado de las restricciones incumplidas
        for(var key in input.validity){
            var status = input.validity[key];
            if(status){
                va.innerHTML += key + ": " + status.toString();
                va.innerHTML += "<br/>";
            }
        }

    } else {
        // Todo correcto.
        // Eliminamos el mensaje y el listado.
        input.setCustomValidity(‘’);
        vm.innerHTML = "";
        va.innerHTML = "";
    }
}

// Recuperamos el elemento con ID status y le asignamos un evento para
// comprobar su validez
var s = document.getElementById("status");
s.addEventListener("invalid", check, true);
```

1.5 USABILIDAD Y ACCESIBILIDAD EN LOS FORMULARIOS

Siempre que se pueda, hay que agrupar la información que se solicita al usuario y ordenarla por relevancia para evitar que se sienta incómodo con la información que se le está solicitando.

Un ejemplo gráfico de esto podría ser es la solicitud de direcciones en donde, por lo general, se solicitan los datos con el siguiente orden: Tipo de Vía, Dirección, Número, Código Postal, Localidad y Provincia.

También es bueno tener presente la sensibilidad de los datos y establecer un orden de petición adecuado. Por ejemplo, en los formularios de registro se suele pedir primero el nombre de usuario, después el email y luego, más tarde, sus datos personales como el nombre completo del usuario. Es una forma de irle guiando, poco a poco, y evitar el abandono por preguntas como ¿por qué me solicitan esto ahora?.

Otra norma que se suele establecer a la hora de solicitar información en los formularios es solicitar primero los campos que sean requeridos y establecer la solicitud de información por pasos, para evitar que los usuarios puedan abrumarse o frustrarse y abandonar.

1.5.1 Autocompletado

La opción de autocompletado es como si se añadiese una capa predictiva sobre el campo. Los campos de los formularios suelen tener la opción de autocompletar para evitar reescribir los valores que ya se habían escrito con anterioridad. Esta opción no se recomienda desactivarla.

El autocompletado también suele referirse a la acción de rellenado causado por otra acción. Una buena praxis es que, si hay campos que pueden ser rellenados de forma automática, se haga. Un ejemplo de ello es el autocompletado de la provincia ya que puede extraerse a partir de la IP del visitante o a través del código postal.

1.5.2 Estructuración y optimización

Una de las prácticas más habituales en el diseño de formularios es que se construyan en formato tabla con varias columnas. Esto puede ser una buena praxis salvo cuando se muestran en un dispositivo móvil ya que, una sobrecarga de información puede hacer que se vuelva ilegible.

En dispositivos de escritorio, no se deben establecer más de tres campos, con sus respectivas etiquetas, por fila, es decir, no debe haber más de seis columnas entre etiquetas y campos.

En dispositivos móviles, con resoluciones menores a 640 píxeles de ancho, los campos deberán organizarse en formato de una única columna, a no ser que sean de un tamaño tal que permita la representación clara sin pérdida de legibilidad.

Otro factor importante es que esté bien armado. Esto se vuelve especialmente importante en dispositivos móviles ya que el espacio de visión está muy reducido.

1.5.3 Capacidades de los dispositivos

Siempre que se pueda, hay que utilizar las capacidades que proporcionan los dispositivos y estudiar si van a ser una mejora o un impedimento. De hecho, hay algunas características que ya se sabe que experiencia de usuario proporcionan.

Por ejemplo, si se está manejando un dispositivo móvil:

- Se debe evitar en la medida de lo posible que el usuario tenga que utilizar el teclado móvil para rellenar formularios.
- Siempre que se pueda, hay que sustituir los campos de tipo texto por botones de opción, verificación o desplegables.
- Los dispositivos móviles soportan gran número de tipos de datos en los formularios. La elección de un color, una fecha, un rango, o introducción de e-mails se hace mucho más sencillo cuando se utilizan los tipos de datos predefinidos.

1.5.4 Combos o desplegables

El único caso dónde el uso de desplegables está justificado es en aquellos casos en los que la respuesta está predefinida y el número de opciones es pequeño. La principal razón de esta afirmación es que, en muchas ocasiones, es más rápido escribir que seleccionar una opción.

El uso de desplegables muy grandes ralentizan la página e incrementa su peso por lo que el tiempo de latencia aumenta y se tarda más tiempo en acceder a los datos.

1.5.5 Número de campos

Está demostrado que el número de campos de los formularios puede crear frustración e inseguridad. Por esta razón, se debe acotar al mínimo posible, es decir, sólo hay que solicitar los campos imprescindibles.

Si el número de campos necesarios es muy extenso se deben dividir en varias páginas (no pestañas) intentando agrupar los campos relacionados y mostrando, primero los requeridos y después los opcionales.

1.5.6 Nombres de campo

Establecer nombres de campos claros y compresibles es fundamental para que el usuario no cometa errores y proporcione la información con más fluidez. No se deben utilizar palabras técnicas, ni hacer preguntas complejas.

Para el diseño de formularios, los campos deben estar incrustados dentro de elementos `LABEL` o estar asociados a ellos mediante el atributo `FOR`. La razón de utilizar esta forma es porque, además de estructurar, ayudan a asegurar la accesibilidad web.

Para facilitar la comprensión de lo que se tiene que introducir o seleccionar, el nombre del campo debe establecerse encima del campo salvo cuando el dato solicitado se muestra a través de casillas de verificación o a través de botones de opción única.

1.5.7 Tipos de datos estándar

Los formularios tienen una gran cantidad de tipos de datos distintos. Cada tipo de dato está definido para un fin concreto y, normalmente, para conseguir una mejor experiencia de usuario.

Cuando se diseñan las interfaces para entornos de escritorio, la cantidad de tipos de datos disponibles se ve mermada por la incompatibilidad que existe entre

los diferentes agentes de usuario. Los tipos de datos que funcionan en uno pueden no funcionar en otro. Por esta razón, en las interfaces de escritorio se tiende a utilizar el tipo texto para muchas de las necesidades.

Si el diseño es para dispositivos móviles, la cosa cambia. Prácticamente todos los tipos de datos que existen están disponibles en los navegadores móviles. Los campos de tipo fecha, por ejemplo, muestran un calendario que ayuda a introducir los datos casi sin esfuerzo.

Siempre que se vaya a diseñar un formulario se debe estudiar la compatibilidad de los tipos de datos en todos los dispositivos dónde se va a poder utilizar y usarlos con inteligencia.

1.5.8 Botones y enlaces

Los botones deben tener apariencia de botón y no ser extravagantes. Un diseño no formal puede confundir a los usuarios y provocarles frustración.

Se debe separar los botones por funcionalidad y/o contexto. Un buen ejemplo es establecer las acciones de "guardar" o "aceptar" en un extremo de la pantalla y las acciones de "volver" o "cancelar" en el otro. Esto evitará que los usuarios cometan errores tales como pinchar por "accidente" en una opción que no era la deseada.

Los botones deben tener un tamaño suficientemente grande para tener un mejor acceso y diferenciarse por su estilo y forma del resto de contenidos. Por ejemplo, los botones de acción como "guardar" o "aceptar" deben tener un estilo diferente a los botones de "volver" o "cancelar" y al resto de enlaces.

Los títulos de acciones como "Ver más", "Seguir leyendo" o "Click aquí" empobrecen la usabilidad y ralentizan el reconocimiento visual de los usuarios ya que pueden perder el contexto de la acción. Además, puede llegar a ser un problema grave de accesibilidad web.

Se debe poder distinguir los diferentes estados de los elementos. Por ejemplo, si se definen estilos diferentes para los estados de un enlace, los usuarios prestarán más atención a los contenidos ya que no perderán el tiempo en realizar reconocimientos visuales para averiguar lo tienen seleccionado o que lo han visitado con anterioridad.

1.5.9 Rangos de valores

Los rangos de valores se pueden utilizar cuando el valor a recuperar no es relevante para el usuario, como pasa con la edad o cuando se solicitan intervalos de datos cortos fácilmente seleccionables a través de un efecto de arrastrar y soltar, como pueden ser las puntuaciones de un valor porcentual.

Los rangos deben tener un tamaño adecuado para ser utilizados con el pulgar si se trata de un dispositivo móvil.

Además, deben tener asociado un campo de entrada de datos de tipo numérico que tenga la misma funcionalidad que un `INPUT` de tipo `NUMBER`. Es decir, que no permita introducir letras o símbolos, que pueda ser incrementado con los cursores de arriba y abajo del teclado o pulsando con el ratón en los símbolos de arriba o abajo, que impida que, este campo numérico, admita valores no contemplados por el rango de valores y que ambos, rango y campo, estén sincronizados si se produce un cambio en el valor de cualquiera de los dos.

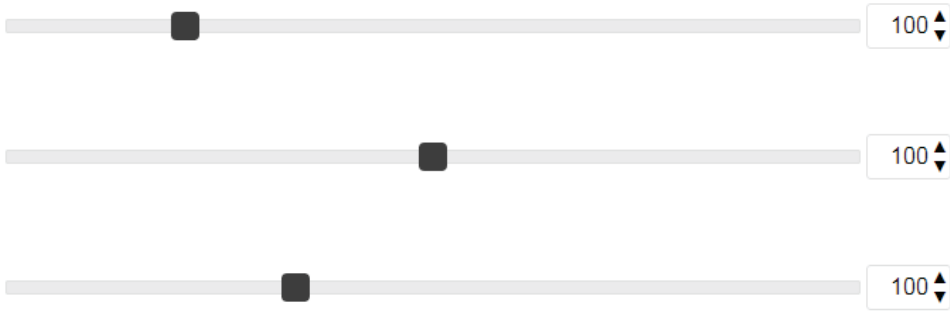


Figura 1.1. Ejemplo control de rango de valores estilizados

1.6 PRÁCTICA 7: PÁGINA DE ACCESO DE "UNIVERES"

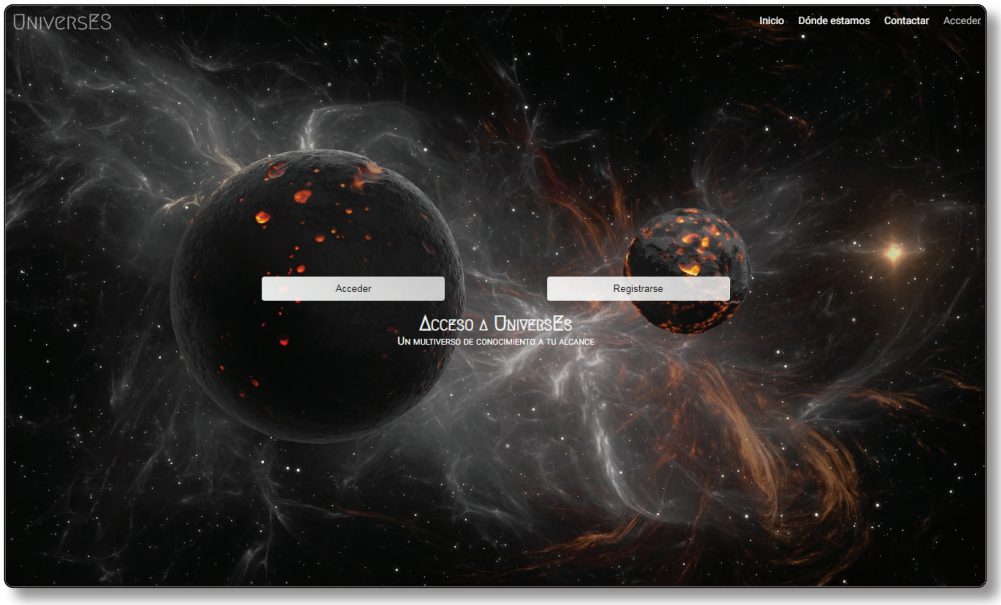
Código del ejemplo

https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-07/pages/login/

Objetivo de la práctica

Diseñar un formulario de acceso y registro combinado.

Resultado



1.6.1 Recursos para hacer la práctica

Recurso

https://pixabay.com/es/photos/la-astronomía-espacio-luna-galaxy-3199541/

Para qué este recurso

Imagen de fondo. La versión que se insertará es la que tiene un tamaño de 1920x1080. Su texto alternativo será el "acceso-a-universES".

En esta página seguiremos un poco la misma línea de diseño que en la página de inicio, es decir, tras la carga aparecerá a pantalla completa sin deformarse y, cuando se muestre uno de los formularios se establecerá a una altura de 250 píxeles.

1.6.2 Resolución

Ya que es el primer contacto con los formularios, lo primero que haremos es una página de acceso a zona privada sencilla. Cabe mencionar que, aunque el HTML y JavaScript utilizados son bastante triviales, el CSS es algo más complicado. No obstante, trataremos de explicar todas las partes con el mayor rigor posible.

Se trata de una página que mostrará una imagen a pantalla completa con un título y los botones de acción de acceso y registro. Cuando el usuario pulse en uno de los dos botones, se reajustará dicha imagen al ancho de la pantalla y mostrará el formulario correspondiente.

El formulario de acceso tendrá los campos nombre de usuario y contraseña, junto con una casilla de verificación para indicar si se debe mantener la sesión iniciada o no. Además, se ofrecerán dos acciones para poder recuperar la contraseña y, por supuesto, poder acceder normalmente.

El formulario de registro tendrá los campos nombre de usuario, contraseña y repetir contraseña (para poder ratificar que la contraseña es correcta), junto con una casilla de verificación para indicar si se está de acuerdo con la Política de Privacidad y Términos de uso o no. Como posibles acciones, sólo se ofrecerá un botón para poder registrarse.

1.6.2.1 EXPLICACIÓN DEL CÓDIGO /PAGES/LOGIN/INDEX.HTML

En lo referente a la configuración del documento y cabecera no vamos a decir nada puesto que únicamente cambia el título del mismo y la clase `ACTIVE` en el menú de navegación y, con respecto al cuerpo del documento, como siempre, definiremos un elemento `MAIN` que contendrá un elemento `ARTICLE` con una cabecera y un elemento agrupador `FIELDSET`.

En la cabecera del artículo (elemento `ARTICLE > HEADER`), como en anteriores ocasiones, definiremos una imagen, un título y un slogan.

En el elemento `FIELDSET` expuesto a continuación de dicha cabecera, estableceremos los dos formularios de acceso y registro con los campos mencionados anteriormente. Cada campo o elemento de formulario irá dentro de un elemento `LABEL`, para ayudar a contextualizar los diferentes tipos de contenido y con todos los atributos pertinentes.

Si nos fijamos en el código de dicho elemento, podremos observar que se utilizan clases que nunca se han definido y algunos estilos en línea para ajustar al comportamiento deseado. Esto último (los estilos en línea) deben usarse con cautela puesto que puede llegar a ser una mala praxis. Esto es, sólo se deben establecer estilos en línea cuando sea estrictamente necesario o cuando la declaración de estilo no vaya a perjudicar la herencia del código.

Por último, sólo nos resta copiar el elemento FOOTER de nuestro documento, que resulta ser exactamente idéntico a todos los anteriores e incluir el código JavaScript localizado en la carpeta JS.

1.6.2.2 CAMBIOS EN EL ARCHIVO STYLES.CSS

En lo referente al CSS, lo primero que realizaremos es la definición de los elementos de formulario, uno a uno. Para todos los elementos INPUT que no sean de tipo CHECKBOX, se les establecerá un estilo común con fondo blanco y sólo con el borde inferior definido. Además, tendrán una altura de 40 píxeles, un estilo de negrita de peso medio y ocuparán el cien por cien del ancho del contenedor.

A los textos asociados a los campos (elementos SPAN de cada LABEL), se les dotará de un efecto de movimiento cuando estén rellenados o enfocados. En otras palabras, cuando el usuario tenga enfocado el elemento INPUT, o haya introducido un valor, el texto proporcionado por los SPAN, se quedará encima del valor que introduzca el usuario. En caso de que esté vacío o desenfocado, aparecerá como si fuese un valor de la propiedad PLACEHOLDER.

Los elementos LABEL tendrán un posicionamiento relativo para poder fijar el posicionamiento absoluto de los SPAN, que son los que utilizaremos como sustituto de la propiedad PLACEHOLDER. Recodemos que, la posición (0,0) de un elemento que posee posicionamiento absoluto, coincide con la posición actual del último elemento padre que tiene posicionamiento relativo.

El elemento SPAN que está contenido dentro del LABEL estará situado a 20 píxeles del borde superior del elemento padre y tendrá un efecto de animación realizado a través de propiedad TRANSITION. Esta propiedad, y otras, se verán más adelante en un capítulo posterior, pero lo que importa en este caso es que, el efecto, provocará un movimiento suave que durará 300 milisegundos modificando la propiedad TOP hasta un valor de CERO.

Los elementos INPUT de tipo CHECKBOX, RADIO y FILE son un tipo especial de elemento de formulario porque, entre otras cosas permiten la personalización a través de CSS mediante los pseudo-elementos BEFORE y AFTER.

En términos generales, la personalización de un elemento de tipo CHECKBOX o RADIO se basa en ocultar el estilo predefinido por los agentes de usuario mediante el pseudo-elemento BEFORE y personalizar el símbolo de verificación a través del pseudo-elemento AFTER.

En nuestro ejemplo, cuando el elemento no esté seleccionado, se aplicará un estilo de fondo blanco con borde gris y un redondeado del borde mediante el pseudo-elemento BEFORE únicamente. Sin embargo, cuando el elemento esté seleccionado se

modificará el fondo a negro a través de la combinación de la pseudo-clase CHECKED y pseudo-elemento BEFORE.

Algo muy similar sucede con el pseudo-elemento AFTER. Por defecto, se definirá una figura que representa el símbolo de verificación mediante técnicas de CSS y que, esencialmente, trata los bordes y juega con la orientación del elemento a través de la propiedad TRANSFORM. Para hacer que aparezca y desaparezca el símbolo de verificación, cuando el elemento no esté seleccionado se le aplicará la propiedad OPACITY con un valor 0 y, cuando lo esté, se le aplicará la propiedad OPACITY con un valor 1.

Ahora vamos con los botones y enlaces. Como ya se definió un estilo de botón en la página de inicio, reutilizaremos parte de ese código. Si nos fijamos en el diseño propuesto para esta página, los botones de ACCEDER y REGISTRARSE no parecen similares a los propuestos en la página de inicio, sin embargo, sólo cambia el color de fondo, si exceptuamos unos pocos ajustes adicionales aplicables al caso actual.

En lo referente a los botones con aspecto de enlace y enlaces en sí, lo único que haremos es definir unos estilos básicos para que no pierda la concordancia con el resto de los elementos sin perder el significado de lo que son. Recordemos que un enlace debe parecer un enlace porque, en caso contrario, los usuarios pueden no saber qué es y frustrarse.

Por fin hemos llegado a la definición de estilos propia de la página que queremos diseñar. En este caso, lo primero que haremos es definir el fondo del documento a negro y establecer los elementos MAIN y ARTICLE al cien por cien del ancho de la pantalla.

En la cabecera de ese elemento ARTICLE configuraremos una imagen para que se ajuste al tamaño de la pantalla, pero cuando el usuario pulse en uno de los botones, se reajuste a una altura de 250 píxeles, sin deformarse. Dichos botones aparecerán centrados en la pantalla y se ocultarán cuando se muestre uno de los formularios asociados.

Al elemento FIELDSET que contiene los formularios, le definiremos un fondo blanco con un borde transparente de 10 píxeles que, por defecto, aparecerá oculto, gracias, en parte, a que la propiedad MAX-HEIGHT contendrá el valor CERO.

Si nos fijamos en el código de dicho elemento, veremos que se hace referencia a unas clases denominadas SHOWING-SIGNIN y SHOWING-SIGNUP que indican cuál es el formulario es el que se desea mostrar y se manipulan mediante una pequeña función en JavaScript que es invocada por el evento ONCLICK de los botones de acción.

Esa función de JavaScript está definida en el archivo SCRIPTS.JS y, básicamente, lo que hace es establecer una clase en el elemento BODY para que se muestre uno u otro formulario a través de CSS.

1.7 PRÁCTICA 8: EXPERIMENTO DE PÁGINA DE CONTACTO DE "UNIVERES"

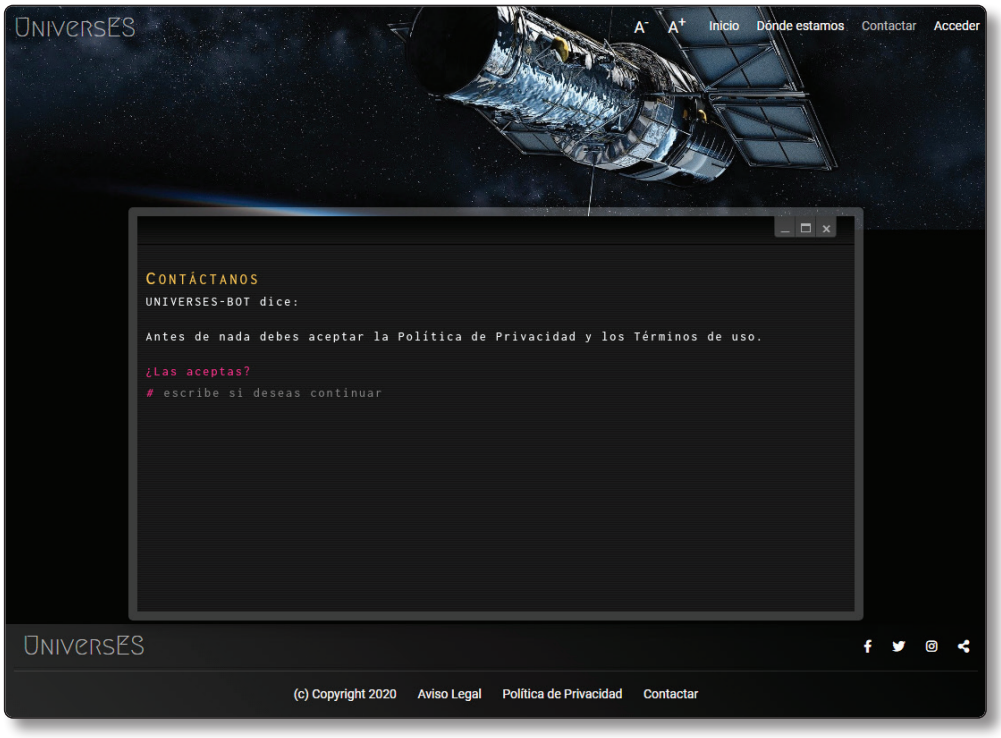
Código del ejemplo

https://github.com/us-ability/contruccion-de-paginas-web/tree/master/practica-08/pages/contactar/

Objetivo de la práctica

Diseñar un formulario de contacto experimental.

Resultado



1.7.1 Recursos para hacer la práctica

Recurso
https://pixabay.com/es/illustrations/telescopio-hubble-universo-1347645/

Para qué este recurso

Esta es la dirección de la imagen para la imagen de cabecera. La versión que se insertará es la que tiene un tamaño de 1920x1080. Su texto alternativo será el "Telescopio-Hubble-cabecera".

En esta página seguiremos un poco la misma línea de diseño que en la página de "Dónde estamos", es decir, la imagen tendrá una altura de 250 píxeles, pero ocupando el cien por cien del ancho de la pantalla sin deformarse.

1.7.2 Resolución

La idea de esta práctica es simular una aplicación parecida a un símbolo de sistema o línea de comandos que ofrecían los terminales de los años 80, aunque algo más actual. Para conseguir este efecto, se diseñará un formulario que irá lanzando preguntas cada vez que se pulse la tecla de retorno de carro.

Primero se solicitará si acepta la política de privacidad y términos de uso. Después, el nombre y el email para realizar el contacto y, finalmente, un campo de texto para que escriba el mensaje.

1.7.2.1 EXPLICACIÓN DEL CÓDIGO /PAGES/CONTACTAR/INDEX.HTML

Al igual que sucede con la práctica anterior, en lo referente a la configuración del documento y cabecera no vamos a decir nada puesto que, únicamente, cambia el título del mismo y la clase `ACTIVE` en el menú de navegación.

Con respecto al cuerpo del documento, como siempre, definiremos un elemento `MAIN` que contendrá un elemento `ARTICLE` con una cabecera y un elemento agrupador `FIELDSET`. En la cabecera del artículo, como en anteriores ocasiones, definiremos una imagen, sin embargo, en esta ocasión no se mostrará ningún título ni slogan.

Lo que sí que cambiará, y bastante, será el contenido del elemento FIELDSET, el cual contendrá un elemento LEGEND con un texto y tres botones y, un elemento FORM con los diferentes elementos necesarios.

Dentro de ese elemento LEGEND colocaremos un elemento de título H1 y un elemento de párrafo como entradillas a la petición de datos. Seguidamente, mostraremos un texto de aviso indicando que, si no se aceptan la política de privacidad y los términos de uso, no se podrá continuar.

Una vez hayamos definido los textos introductorios, se deberán establecer una serie de bloques o sentencias que conformarán cada una de las distintas entradas de datos. Cada uno de estos bloques contendrá los siguientes elementos:

- Un elemento LABEL con un atributo FOR que identificará a qué INPUT está asociado.
- Un elemento I que servirá como decorador de línea de comandos.
- El elemento INPUT que será utilizado para la recogida de datos
- Un elemento SPAN que servirá como contenedor de mensajes de error.

Además, cada uno de los elementos INPUT tendrá una serie de atributos que lo especificarán su cometido y nos ayudarán a validar la entrada de datos. En general, cada uno de ellos podrá contener:

- Un atributo ID para identificar el elemento de formulario y asociarlo a la etiqueta LABEL.
- Un atributo NAME para que pueda ser enviado y recibido por el servidor.
- Un atributo REQUIRED para indicar que es obligatorio y validable.
- Un atributo PLACEHOLDER para ayudar al usuario a contextualizar la entrada de datos y facilitar su comprensión, lo que evitará errores involuntarios o de entendimiento.
- Un atributo ONKEYDOWN para controlar cuando se pulsa la tecla ENTER o retorno de carro.
- Un atributo MINLENGTH para indicar la longitud mínima de caracteres.
- Un atributo MAXLENGTH para indicar la longitud máxima de caracteres.
- Un atributo PATTERN que se utilizará para validar que se insertan los valores correctos.
- El atributo AUTOFOCUS para indicar que se establezca el cursor en ese elemento tras la carga de la página.

Como se podrá apreciar en el código del formulario, se define una estructura cada vez que se desea solicitar una información y se caracteriza porque tiene asignada la clase SENTENCE. Algunas de estas estructuras presentan la clase NEXT que, además de ser útiles a la hora manipular los distintos bloques en JavaScript, identificarán

los siguientes pasos por los que se pasará un usuario cuando entre en la página y permanecerán ocultos hasta el momento de utilizarlos.

Seguidamente, copiamos el elemento FOOTER de documentos o prácticas anteriores y el código JavaScript genérico. En este punto, y sólo para este documento, añadiremos una etiqueta SCRIPT que contendrá el JavaScript necesario para su correcto funcionamiento.

1.7.2.2 CÓDIGO ARCHIVO /JS/CONTACT.JS

Una vez que hemos declarado el HTML de la página, es hora de declarar el código JavaScript necesario para que se produzca una buena experiencia cuando el usuario interactúe con la página.

Primero declararemos una función denominada WHENKEYDOWN que controlará la tecla de retorno de carro o aceptar y que, en JavaScript, se corresponde con el código 13 del sistema de codificación ASCII.

A continuación, se comprobará que la entrada de datos es válida y, de ser así, se recuperará el siguiente bloque a mostrar a través de la clase NEXT. Este proceso, también nos permitirá verificar si el elemento que pulsó la tecla de aceptar es el elemento EMAIL y, si es así, en el elemento siguiente a él, se sustituirá la palabra clave @NAME por el nombre solicitado con anterioridad. Además, si todo ha ido bien, se almacenarán localmente todos los datos solicitados a través de la API LOCALSTORAGE de HTML5 para no tener que pedirle los datos todas las veces que entre.

Ahora que tenemos controlado el mostrado de sentencias y la acción de aceptar, definiremos un método llamado SEND que será quién envíe al servidor el mensaje por el que el usuario desea contactar con nosotros. No obstante, como esto es una maqueta, sólo mostraremos un mensaje de confirmación y crearemos otra entrada de datos de forma dinámica mediante la función APPENDNEWSENTENCE.

Esta función APPENDNEWSENTENCE, será quién replique el bloque de la última sentencia y vaya modificando los atributos y valores para que se pueda seguir utilizando la página.

Como hemos visto antes, se ha realizado una llamada a una función denominada SAVEDATA. Esta función es quién almacena en la LOCALSTORAGE la información del usuario para reutilizarla más tarde.

Antes de finalizar con este script, definiremos una función o método denominado TOGGLEMAXIMIZE y un manejador de eventos para cuando se realiza un cambio de tamaño en la resolución de pantalla disponible a través de WINDOW.ONRESIZE. Lo que nos permitirán estos métodos es gestionar la API SCREEN de HTML5 para cambiar el modo de presentación del contenido de la página. Esto es, a pantalla completa sin la barra de direcciones, menús, marcadores ni botones o, en modo normal, como es lo habitual.

Finalmente, dado que guardamos la información en local, definiremos un proceso de recuperación tras la carga de la página. Este proceso lo que realizará es la ocultación de los campos ya almacenados, personalizar la entrada con el nombre y poner el cursor en la sentencia que solicita el motivo del contacto.

1.7.2.3 CAMBIOS EN EL ARCHIVO STYLES.CSS

En lo referente al CSS, lo primero que realizaremos es la definición de unos estilos generales para el documento. Dado que esta página es especial, modificaremos los estilos asociados al elemento `BODY`, al elemento `MAIN` y al elemento `ARTICLE`. No obstante, no serán los únicos elementos a modificar o retocar, también deberemos actualizar los estilos aplicados a la imagen de la cabecera y de los elementos de párrafo.

Al igual que en prácticas anteriores, el contenido del artículo se estableció a través de un elemento `FIELDSET`. Este elemento será tratado, en esta práctica, de manera especial porque será quién proporcione la apariencia de "terminal". Además, para que nos recuerde a los monitores de los 80, se le dotará de un efecto de escaneado vertical a través de la propiedad `BEFORE`.

Si nos fijamos en el código, veremos que para hacer el efecto del escaneado y del fondo del terminal se ha recurrido a gradientes, uno repetitivo y otro no. Además, para que se ajuste a la mayor parte de resoluciones de pantalla se han definido varias consultas de medios que establecen el ancho en función de la altura.

Ahora, por comodidad, lo que haremos es establecer unos estilos generales para todos los elementos que estén dentro de nuestro `FIELDSET`. Entre estos estilos, estableceremos un color por defecto, un formato de letra con peso y tamaño específicos y una separación entre caracteres similar a la utilizada en las resoluciones de entonces. Además, si el dispositivo es móvil, le aumentaremos el tamaño de letra para mejorar su lectura y legibilidad.

Como se puede observar, en la regla `.CONTACT FIELDSET *`, se utiliza una fuente llamada "Inconsolata", por lo que se deberá añadir a nuestra hoja de estilos al principio de la misma. A continuación, se muestran las fuentes que deberían estar declaradas después de estos últimos cambios.

Al igual que los programas de la época, definiremos una botonera en la parte superior con las acciones de minimizar, maximizar y cerrar. Todas estas acciones se definirán dentro del elemento `LEGEND` del `FIELDSET`.

Por último, sólo nos queda definir los estilos del `FIELDSET`, incluyendo sus elementos internos, cuando está en modo maximizado. Es decir, como el modo maximizado ocultará la cabecera y pie de la página, le pondremos el identificador del sitio web en la leyenda, cambiaremos los anchos y altos del `FIELDSET` para que se ajuste al cien por cien, y definiremos un nuevo estilo para el botón `TOGGLE` ya que ahora está maximizado y el símbolo de restaurar muy diferente.