

HTML. Tema 9. Formularios.

[1. Introducción](#)

[2. ¿Qué son los formularios web?](#)

[3. Controles básicos de forma nativa](#)

[3.1. El elemento <label>](#)

[3.1.1. ¡También se puede hacer clic en las etiquetas!](#)

[3.2. Campos de entrada de texto](#)

[3.2.1. Campos de texto de una sola línea](#)

[3.2.1.1. Campo de contraseña](#)

[3.2.2. Campos de texto de varias líneas](#)

[3.2.3. Valores predeterminados en campos de texto](#)

[3.3. Elementos que se pueden marcar: casillas de verificación y botones de opción](#)

[3.3.1. Caja](#)

[3.3.2. Boton de radio](#)

[3.4. El elemento <select>](#)

[3.5. Selector de archivos](#)

[3.6. Botones](#)

[3.7. Ejercicio propuesto: controles nativos](#)

[4. Tipos de entrada HTML5](#)

[4.1. Dirección de correo electrónico vejez](#)

[4.2. Campo URL](#)

[4.3. Número de teléfono vejez](#)

[4.4. Campo numérico](#)

[4.5. Controles deslizantes](#)

[4.6. Selectores de fecha y hora](#)

[4.7. Control del selector de color](#)

[4.8. Campo de búsqueda](#)

[4.9. Ejercicio propuesto: tipos de entrada HTML5](#)

[5. Tu primera forma "real"](#)

[5.1. Diseñando tu formulario](#)

[5.2. Implementando tu formulario](#)

[5.3. El elemento <form>](#)

[5.4. Los elementos <label>, <input> y <textarea>](#)

[5.5. El elemento <button>](#)

[5.6. Envío de formdata a su servidor web](#)

[5.6.1. Validación de formularios del lado del cliente](#)

[5.6.1.1. Otras lecturas](#)

[5.6.2. En el lado del servidor: recuperar los datos](#)

[5.6.3. El método GET](#)

[5.6.3.1. Ejercicio propuesto: formulario de contacto](#)

[5.6.3.2. Ejercicio propuesto: formulario de contacto completo](#)

[5.6.3.3. Ejercicio propuesto: formulario de saludos](#)

[5.6.3.4. Ejercicio propuesto: formulario completo de saludos](#)

[5.6.4. El método POST](#)

[5.6.4.1. Ejercicio propuesto: formulario de inicio de sesión](#)

[5.6.4.2. Ejercicio propuesto: formulario de inicio de sesión completo](#)

1. Introducción

Esta unidad proporciona algunas instrucciones y ejemplos que le ayudarán a aprender los conceptos básicos de los formularios web. Los formularios web son una herramienta muy poderosa para interactuar con los usuarios; por lo general, se utilizan para recopilar datos de los usuarios o para permitirles controlar una interfaz de usuario. Sin embargo, por razones históricas y técnicas, no siempre es obvio cómo utilizarlos en todo su potencial. En las secciones que se enumeran a continuación, cubriremos todos los aspectos esenciales de los formularios web, incluido el marcado de su estructura HTML, la validación de los datos del formulario y el envío de datos al servidor.

2. ¿Qué son los formularios web?

Formularios web son uno de los principales puntos de interacción entre un usuario y un sitio web o aplicación. Los formularios permiten a los usuarios ingresar datos, que generalmente se envían a un servidor web para su procesamiento y almacenamiento, o se usan en el lado del cliente para actualizar inmediatamente la interfaz de alguna manera (por ejemplo, agregar otro elemento a una lista, o mostrar u ocultar un Función de interfaz de usuario).

El HTML de un formulario web se compone de uno o más **controles de formulario** (a veces llamado **widgets**), además de algunos elementos adicionales para ayudar a estructurar la forma general; a menudo se los denomina **Formularios HTML**.

Los controles pueden ser campos de texto de una o varias líneas, cuadros desplegables, botones, casillas de verificación o botones de radio, y en su mayoría se crean utilizando el `< entrada >` elemento, aunque también hay algunos otros elementos sobre los que aprender.

Los controles de formulario también se pueden programar para imponer formatos o valores específicos que se deben ingresar (**validación del formulario**), y junto con etiquetas de texto que describen su propósito para usuarios ciegos y videntes.

3. Controles básicos de forma nativa

En las siguientes secciones, marcaremos varios ejemplos de formularios web funcionales, utilizando algunos controles de formulario y elementos estructurales comunes, y centrándonos en las mejores prácticas de accesibilidad. Ahora veremos la funcionalidad de los diferentes controles de formulario, o widgets, en detalle, estudiando todos los diferentes opciones disponibles para recopilar diferentes tipos de datos. En esta sección en particular, veremos el conjunto original de controles de formulario, disponible en todos los navegadores desde los primeros días de la web.

3.1. El elemento <label>

los `< etiqueta >` elemento es la forma formal de definir una etiqueta para un widget de formulario HTML. Este es el elemento más importante si desea crear formularios accesibles. Cuando se implementan correctamente, los lectores de pantalla leerán la etiqueta de un elemento de formulario junto con las instrucciones relacionadas, además de ser útiles para los usuarios videntes. Tome este ejemplo, donde anidamos el control de formulario dentro del `< etiqueta >`, asociándolo implícitamente:

```
1. < etiqueta >
2.     Nombre: < entrada tipo = " texto" nombre = " nombre" /> </ etiqueta >
3.
```

Con el `< etiqueta >` asociado correctamente con el `< entrada >` un lector de pantalla leerá algo como "Nombre, editar texto". Si no hay ninguna etiqueta, o si el control de formulario no está asociado implícita o explícitamente con una etiqueta, un lector de pantalla leerá algo como "Editar texto en blanco", lo cual no es muy útil en absoluto.

3. 1. 1. ¡Las etiquetas también son clicables!

Otra ventaja de configurar correctamente las etiquetas es que puede hacer clic o tocar la etiqueta para activar el widget correspondiente. Esto es útil para controles como entradas de texto, donde puede hacer clic en la etiqueta así como en la entrada para enfocarla, pero es especialmente útil para botones de radio y casillas de verificación. El área de impacto de dicho control puede ser muy pequeña, por lo que es útil que sea lo más fácil de activar posible.

Por ejemplo, al hacer clic en las etiquetas "Me gusta la cereza" o "Me gusta el plátano" en el ejemplo siguiente, se alternará el estado seleccionado de la *Cereza* o *plátano* casillas de verificación respectivamente:

```
1. < etiqueta >
2.   < entrada tipo = " caja" nombre = " Cereza" valor = " Cereza" />
3.   Me gusta la cereza
4. </ etiqueta > < br />
5. < etiqueta >
6.   < entrada tipo = " caja" nombre = " plátano" valor = " plátano" />
7.   me gusta el platano
8. </ etiqueta > < br />
```

3.2. Campos de entrada de texto

Texto < [entrada](#) > Los campos son los widgets de formulario más básicos. Son una forma muy conveniente de permitir que el usuario ingresar cualquier tipo de datos porque pueden tomar muchas formas diferentes dependiendo de su tipo valor de atributo. Se utiliza para crear la mayoría de los tipos de widgets de formulario, incluidos campos de texto de una sola línea, controles de fecha y hora, controles sin entrada de texto como casillas de verificación, botones de opción y selectores de color, y también botones.

Todos los controles de texto básicos comparten algunos comportamientos comunes:

- Pueden marcarse como [necesario](#) , para especificar que es necesario completar un campo de formulario antes de poder enviarlo.
- Pueden marcarse como [solo lectura](#) (el usuario no puede modificar el valor de entrada pero aún se envía con el resto de los datos del formulario) o [discapacitado](#) (el valor de entrada no se puede modificar y nunca se envía con el resto de los datos del formulario).
- Pueden tener un [marcador de posición](#) . Este es el texto que aparece dentro del cuadro de entrada de texto que debe usarse para describir brevemente el propósito del cuadro.
- Pueden estar restringidos en [Talla](#) (el tamaño físico de la caja) y [longitud mínima](#) y [longitud máxima](#) (el número mínimo y máximo de caracteres que se pueden ingresar en el cuadro). Pueden beneficiarse de la revisión ortográfica (utilizando el [corrector ortográfico](#) atributo),
- si el navegador lo admite.

Tenga en cuenta que los campos de texto de los formularios HTML son simples controles de entrada de texto sin formato. Esto significa que no puede utilizarlos para realizar [edición rica](#) (negrita, cursiva, etc.). [Todos los editores](#) de texto enriquecido que encontrará son widgets personalizados creados con HTML, CSS y JavaScript.

3. 2. 1. Campos de texto singl el ine

Se crea un campo de texto de una sola línea `< entrada >` elemento cuyo tipo el valor del atributo se establece en `texto` , o omitiendo el tipo atributo en conjunto `texto` es el valor predeterminado). El valor `texto` para este atributo es también el valor de reserva si el valor que especifica para el tipo El navegador desconoce el atributo (por ejemplo, si especifica `tipo = "color"` y el navegador no admite selectores de color nativos).

Veamos este ejemplo usando un par de campos de texto de una sola línea:

```
1. < etiqueta >
2.     Nombre (de 5 a 10 caracteres):
3.     < entrada tipo = " texto" nombre = " nombre" necesario
4.         minlength = " 5 " maxlength = " 10 " tamaño = " 15"
5.         placeholder = " por ejemplo, Fernando " >
6. </ etiqueta > < br />
7. < etiqueta >
8.     Comentario:
9.     < entrada tipo = " texto" nombre = " comentario" necesario
10.         placeholder = " por ejemplo, me gusta este ejemplo " >
11. </ etiqueta > < br />
```

HTML5 mejoró el campo de texto de una sola línea original básico agregando valores especiales para el tipo atributo que impone restricciones de validación específicas y otras características, por ejemplo, específicas para ingresar correos electrónicos, URL o números. Los cubriremos en una sección a continuación (Los tipos de entrada HTML5).

3.2.1.1. Campo de contraseña

Uno de los tipos de entrada originales fue el contraseña tipo de campo de texto:

```
1. < etiqueta >
2.     Contraseña: < entrada tipo = " contraseña" nombre = " contraseña" >
3. </ etiqueta >
```

los contraseña value no agrega ninguna restricción especial al texto ingresado, pero oculta el valor ingresado en el campo (por ejemplo, con puntos o asteriscos) por lo que otros no pueden leerlo fácilmente.

Tenga en cuenta que esto es solo una característica de la interfaz de usuario; a menos que envíe su formulario de forma segura, se enviará en texto sin formato, lo cual es malo para la seguridad: una parte malintencionada podría interceptar sus datos y robar contraseñas, detalles de tarjetas de crédito o cualquier otra cosa que haya enviado. La mejor manera de proteger a los usuarios de esto es alojar cualquier página que incluya formularios a través de una conexión segura (es decir, en un `https://...` dirección), por lo que los datos se cifran antes de enviarlos.

Los navegadores reconocen las implicaciones de seguridad de enviar datos de formulario a través de una conexión insegura y tienen advertencias para disuadir a los usuarios de utilizar formularios inseguros. Para obtener más información sobre lo que implementa Firefox, consulte [Contraseñas inseguras](#).

3. 2. 2. Campos de texto múltiples

El HTML `<textarea>` El elemento representa un control de edición de texto sin formato de varias líneas, útil cuando desea permitir que los usuarios ingresen una cantidad considerable de texto de forma libre, por ejemplo, un comentario en una revisión o formulario de comentarios:

```
1. <etiqueta> Cuéntanos tu historia:
2.   <textarea nombre = " historia" filas = " 5 " >
3.     Era una noche oscura y tormentosa...
4.   </textarea>
5. </etiqueta>
```

Puedes usar el `filas` y `cols` atributos para especificar un tamaño exacto para el `<textarea>` tomar. A veces, configurarlos es una buena idea para mantener la coherencia, ya que los valores predeterminados del navegador pueden diferir. También estamos usando un contenido predeterminado (ingresado entre las etiquetas de apertura y cierre), ya que `<textarea>` no es compatible con el valor atributo.

los `<textarea>` El elemento también acepta varios atributos comunes para formar `<entrada>` elemento, tal como `autocompletar`, `autoenfocar`, `discapacitado`, `marcador de posición`, `solo lectura`, y `necesario`.

3. 2. 3. Valores predeterminados en los campos de texto

Note que el `<entrada>` La etiqueta es un elemento vacío, lo que significa que no necesita una etiqueta de cierre. los `<textarea>` sin embargo, el elemento debe cerrarse con la etiqueta final adecuada. Esto tiene un impacto en un

característica específica de los formularios: la forma en que define el valor predeterminado. Para definir el valor predeterminado de un

< **entrada** > elemento tienes que usar el valor atributo como este:

1. < **entrada** tipo = " texto" valor = " de forma predeterminada, este elemento se rellena con este texto " >

Por otro lado, si desea definir un valor predeterminado para un < **textarea** > , lo pones entre las etiquetas de apertura y cierre del < **textarea** > elemento, así:

1. < **textarea** >
2. por defecto, este elemento se rellena con este texto
3. </ **textarea** >

3.3. Elementos que se pueden marcar: casillas de verificación y botones de opción

Los elementos marcables son controles cuyo estado se puede cambiar haciendo clic en ellos o en sus etiquetas asociadas. Hay dos tipos de elementos que se pueden marcar: la casilla de verificación y el botón de opción. Ambos usan el comprobado atributo para indicar si el widget está marcado por defecto o no.

Vale la pena señalar que estos widgets no se comportan exactamente como otros formwidgets. Para la mayoría de los widgets de formulario, una vez enviado el formulario, todos los widgets que tienen un nombre se envían los atributos, incluso si no se ha llenado ningún valor. En el caso de elementos comprobables, sus valores se envían solo si están marcados. Si no se marcan, no se envía nada, ni siquiera su nombre. Si están marcados pero no tienen ningún valor, el nombre se envía con un valor de *en*.

3. 3. 1. Caja

Se crea una casilla de verificación utilizando el < **entrada** > elemento con un tipo atributo establecido en el valor [caja](#) . Elementos de tipo caja se representan de forma predeterminada como casillas que están marcadas (marcadas) cuando se activan, como puede ver en un formulario oficial en papel del gobierno. La apariencia exacta depende de la configuración del sistema operativo bajo el cual se ejecuta el navegador. Generalmente es un cuadrado pero puede tener esquinas redondeadas. Una casilla de verificación le permite seleccionar valores únicos para enviarlos en un formulario (o no).

Veamos y probemos un ejemplo muy simple:

```
1. < etiqueta >
2.   < entrada tipo = " caja" nombre = " zanahorias" valor = " zanahorias" comprobado />
3.   ¿Te gustan las zanahorias?
4. </ etiqueta >
```

Incluyendo el comprobado hace que la casilla de verificación se marque automáticamente cuando se carga la página. Al hacer clic en la casilla de verificación o en su etiqueta asociada, la casilla de verificación se activa y desactiva.

Debido a la naturaleza on-o de las casillas de verificación, la casilla de verificación se considera un botón de alternancia. Muchos desarrolladores y diseñadores están ampliando el estilo de las casillas de verificación predeterminadas para crear botones que parecen interruptores de palanca.

3. 3. 2. Boton de radio

Un botón de radio se crea usando el `< entrada >` elemento con su tipo atributo establecido en el valor `radio`. Elementos de tipo radio se utilizan generalmente en **grupos de radio** colecciones de botones de opción que describen un conjunto de opciones relacionadas). Solo se puede seleccionar un botón de opción en un grupo determinado al mismo tiempo. Los botones de opción suelen representarse como pequeños círculos, que se rellenan o resaltan cuando se seleccionan.

Veamos un ejemplo simple que contiene varios botones de opción y cómo un navegador puede representarlo:

```
1. ¿Cuál es tu comida favorita? < br />
2. < etiqueta >
3.   < entrada tipo = " radio" nombre = " comida" valor = " sopa" comprobado /> Sopa
4. </ etiqueta > < br />
5. < etiqueta >
6.   < entrada tipo = " radio" nombre = " comida" valor = " curry" /> Curry
7. </ etiqueta > < br />
8. < etiqueta >
9.   < entrada tipo = " radio" nombre = " comida" valor = " Pizza" /> Pizza
10. </ etiqueta > < br />
```

Como se vio arriba, se pueden unir varios botones de radio. Si comparten el mismo valor por su nombre atributo, se considerará que están en el mismo grupo de botones. Solo se puede marcar un botón de un grupo determinado a la vez; esto significa que cuando uno de ellos está marcado, todos los demás se desmarcan automáticamente. Cuando se envía el formulario, solo se envía el valor del botón de opción marcado. Si no se marca ninguno de ellos, se considera que todo el grupo de botones de opción está en un estado desconocido y no

El valor se envía con el formulario. Una vez que se marca uno de los botones de opción en un grupo de botones con el mismo nombre, el usuario no puede desmarcar todos los botones sin restablecer el formulario.

3.4. El elemento <select>

El HTML < **Seleccione** > elemento representa un control que proporciona un menú de opciones. Por ejemplo:

```
1. < etiqueta > Elige la mascota que más te guste:
2.   < Seleccione nombre = " mascotas" id = " seleccionar mascota " >
3.     < opción valor = "" > - Por favor, elija una opción-- </ opción >
4.     < opción valor = " perro" > Perro </ opción >
5.     < opción valor = " gato" > Gato </ opción >
6.     < opción valor = " hámster" > Hámster </ opción >
7.     < opción valor = " loro" > Loro </ opción >
8.     < opción valor = " araña" > Araña </ opción >
9.     < opción valor = " pez de colores" > Pez de colores </ opción >
10.   </ Seleccione >
11. </ etiqueta >
```

El ejemplo anterior muestra los típicos < **Seleccione** > uso. Está asociado con un < **etiqueta** > con fines de accesibilidad, así como nombre atributo para representar el nombre de los datos asociados enviados al servidor. Cada opción de menú está definida por un < **opción** > elemento anidado dentro del < **Seleccione** > .

Cada < **opción** > el elemento debe tener un valor atributo que contiene el valor de los datos para enviar al servidor cuando se selecciona esa opción. Si no se incluye ningún atributo de valor, el valor predeterminado es el texto contenido dentro del elemento. Puede incluir un seleccionado atributo en un elemento para que se seleccione por defecto cuando se carga la página por primera vez.

los < **Seleccione** > El elemento tiene algunos atributos únicos que puede utilizar para controlarlo, como múltiple para especificar si se pueden seleccionar varias opciones, y Talla para especificar cuántas opciones deben mostrarse a la vez. También acepta la mayoría de los atributos de entrada de formularios generales, como necesario ,
discapacitado , autoenfoque , etc.

3.5. Selector de archivos

Hay una última < **entrada** > tipo que nos llegó en HTML temprano: el tipo de entrada de archivo. Los formularios pueden enviar archivos a un servidor (esta acción específica también se detalla en el [Envío de datos de formulario](#) artículo). El widget selector de archivos puede usarse para elegir uno o más archivos para enviar.

Para crear un [widget de selector de archivos](#), puedes usar el < **entrada** > elemento con su tipo atributo establecido en archivo

. Los tipos de archivos que se aceptan pueden restringirse mediante la [aceptar](#) atributo. Además, si desea permitir que el usuario elija más de un archivo, puede hacerlo agregando el [múltiple](#) atributo.

En el siguiente ejemplo, se crea un selector de archivos para solicitar archivos de imágenes gráficas. El usuario puede seleccionar varios archivos en este caso:

```
1. < entrada tipo = " archivo" nombre = " archivo" aceptar = " imagen/*" múltiple />
```

En algunos dispositivos móviles, el selector de archivos puede acceder a fotos, videos y audio capturados directamente por la cámara y el micrófono del dispositivo agregando información de captura al aceptar atributo así:

```
1. < entrada tipo = " archivo" aceptar = " imagen / *; captura = cámara " >
2. < entrada tipo = " archivo" aceptar = " video / *; capture = videocámara " >
3. < entrada tipo = " archivo" aceptar = " audio / *; captura = micrófono " >
```

3.6. Botones

El HTML < **botón** > elemento representa un botón en el que se puede hacer clic, que se utiliza para enviar formularios o en cualquier lugar de un documento para una funcionalidad de botón estándar y accesible. De forma predeterminada, los botones HTML se presentan en un estilo que se asemeja a la plataforma en la que se ejecuta el navegador, pero puede cambiar la apariencia de los botones con CSS.

El comportamiento predeterminado del botón se puede cambiar con el tipo atributo. Los posibles valores son:

- **enviar** : El botón envía los datos del formulario al servidor. Este es el valor predeterminado si el atributo no se especifica para los botones asociados con el formulario o si el atributo contiene un valor vacío o no válido.

- Reiniciar : El botón restablece todos los controles a sus valores iniciales. Debe usarlo solo cuando sea necesario, ya que este comportamiento tiende a molestar a los usuarios.
- botón : El botón no tiene un comportamiento predeterminado y no hace nada cuando se presiona de manera predeterminada. Puede hacer que los scripts del lado del cliente escuchen los eventos del elemento, que se activan cuando ocurren los eventos.

Veamos todo tipo de botones con un ejemplo sencillo:

```
1. < pag >
2.   < etiqueta > Ingrese su comentario: < entrada tipo = " texto" nombre = " comentario" necesario /> </ etiqueta >
3.
4. < pag >
5.   < botón tipo = " enviar" > Este es un botón de enviar </ botón >
6.
7. < pag >
8.   < botón tipo = " Reiniciar" > Este es un botón de reinicio </ botón >
9.
10. < pag >
11.   < botón tipo = " botón" > Este es un simple botón </ botón >
12. </ pag >
```

Como puede ver en los ejemplos, < botón > Los elementos le permiten usar HTML en su contenido, que se inserta entre la apertura y el cierre. < botón > etiquetas. < entrada > los elementos, por otro lado, son elementos vacíos; su contenido mostrado se inserta dentro del valor atributo y, por lo tanto, solo acepta texto sin formato como contenido.

3.7. Ejercicio propuesto: controles nativos

Cree una página web para mostrar muestras de todos los elementos de entrada en esta sección: texto de una sola línea y de varias líneas, contraseña, casillas de verificación y botones de opción, seleccionar y selector de archivos. Debe incluir al menos dos ejemplos de cada uno de ellos. Tiene que usar párrafos y etiquetas, y también el atributo "requerido" y todas las restricciones de campo necesarias deben establecerse para todos ellos. Verifique el resultado en su navegador y no olvide incluir todas las etiquetas HTML básicas y validar su código. Finalmente, sube el código a tu dominio y verifica el resultado en tu teléfono móvil.

- Pon todas las etiquetas dentro de un `< formar >` contenedor y use un botón de envío para que pueda verificar que los campos están validados correctamente:

```

1.  < formar >
2.    < pag > < etiqueta >
3.      Nombre: < entrada tipo = " texto" nombre = " nombre" necesario /> </ etiqueta > </ pag >
4.
5.    < pag > < etiqueta >
6.      Apellido: < entrada tipo = " texto" nombre = " apellido" necesario /> </ etiqueta > </ pag >
7.
8.    < pag > < etiqueta >
9.      Contraseña: < entrada tipo = " contraseña" nombre = " contraseña1 " necesario /> </ etiqueta > </ pag >
10.
11.   < pag > < etiqueta >
12.     Repita su contraseña: < entrada tipo = " contraseña" nombre = " contraseña2 " necesario /> </ etiqueta > </ pag >
13.
14.     . . .
15.   < pag > < botón > Enviar </ botón > </ pag >
dieciséis </ formar >

```

4. Tipos de entrada HTML5

En la sección anterior miramos el `< entrada >` elemento, cubriendo los valores originales del tipo

atributo disponible desde los primeros días de HTML. Ahora veremos en detalle la funcionalidad de los controles de formulario más nuevos, incluidos algunos tipos de entrada nuevos, que se agregaron en HTML5 para permitir la recopilación de tipos específicos de datos.

4.1. Campo de dirección de correo electrónico

Este tipo de campo se establece mediante el valor correo electrónico Para el [tipo](#) atributo:

```

1.  < etiqueta >
2.    Introduzca un correo electrónico válido:
3.    < entrada tipo = " correo electrónico" nombre = " correo electrónico" placeholder = " por ejemplo, test@test.com " necesario /> </ etiqueta >
4.

```

Cuando esto tipo se utiliza, el usuario debe escribir una dirección de correo electrónico válida en el campo. Cualquier otro contenido hace que el navegador muestre un error cuando se envía el formulario. Puedes ver esto en acción aquí:

También puede utilizar el [multiple](#) atributo en combinación con el correo electrónico tipo de entrada para permitir que se ingresen varias direcciones de correo electrónico en la misma entrada (separadas por comas):

```
1. < etiqueta >
2.      Varios correos electrónicos: < entrada tipo = " correo electrónico" nombre = " correos electrónicos " multiple /> </ etiqueta >
3.
```

En algunos dispositivos (en particular, dispositivos táctiles con teclados dinámicos como teléfonos inteligentes) se puede presentar un teclado virtual diferente que sea más adecuado para ingresar direcciones de correo electrónico, incluido el @ llave. Esta es otra buena razón para usar estos nuevos tipos de entrada, mejorando la experiencia del usuario para los usuarios de estos dispositivos.

4.2. Campo URL

Se puede crear un tipo especial de campo para ingresar URL usando el valor url Para el [tipo](#) atributo:

```
1. < etiqueta >
2.      Introducir URL:
3.      < entrada tipo = " url " nombre = " url " placeholder = " por ejemplo, https: // ... " necesario /> </ etiqueta >
4.
```

Agrega restricciones de validación especiales al campo. El navegador informará un error si no hay protocolo (como http: se ingresa, o si la URL está mal formada. Puedes ver esto en acción aquí:

En dispositivos con teclados dinámicos, el teclado predeterminado a menudo mostrará algunos o todos los dos puntos, el punto y la barra inclinada como teclas predeterminadas.

4.3. Campo de número de teléfono

Se puede crear un campo especial para rellenar números de teléfono usando tel como el valor de la [tipo](#) atributo:

1. `< etiqueta >`
2. Ingresar número telefónico:
3. `< entrada tipo = " tel " nombre = " tel " placeholder = " por ejemplo, 123 456 789 " /> </ etiqueta >`
- 4.

Cuando se accede a través de un dispositivo táctil con un teclado dinámico, la mayoría de los dispositivos mostrarán un teclado numérico cuando tipo = "tel" se encuentra, lo que significa que este tipo es útil siempre que un teclado numérico sea útil, y no solo tiene que usarse para números de teléfono.

Debido a la amplia variedad de formatos de números de teléfono en todo el mundo, este tipo de campo no impone ninguna restricción sobre el valor ingresado por un usuario (esto significa que puede incluir letras, etc.).

4.4. Campo numérico

Los controles para ingresar números se pueden crear con un `< entrada tipo = " número" >` Este control se parece a un campo de texto, pero solo permite números de punto flotante y, por lo general, proporciona botones en forma de ruleta para aumentar y disminuir el valor del control. En dispositivos con teclados dinámicos, generalmente se muestra el teclado numérico.

Con el número tipo de entrada, puede restringir los valores mínimos y máximos permitidos [min](#) y [max](#) atributos. También puede utilizar el paso atributo para establecer el incremento, aumento y disminución causado al presionar los botones giratorios. De forma predeterminada, el tipo de entrada de número solo se valida si el número es un entero. Para permitir números de acento, especifique paso = "cualquiera" Si se omite, el

paso el valor predeterminado es 1 , lo que significa que solo los números enteros son válidos.

Veamos algunos ejemplos. El primero a continuación crea un control numérico cuyo valor está restringido a cualquier valor entre 1 y 10 , y cuyos botones de aumento y disminución cambian su valor en 2 :

1. `< entrada tipo = " número" nombre = " años" min = " 1 " max = " 10 " paso = " 2 " valor = " 1 " />`

El segundo crea un control numérico cuyo valor está restringido a cualquier valor entre 0 y 1 inclusive, y cuyos botones de aumento y disminución cambian su valor por 0. 01 :

1. `< entrada tipo = " número" nombre = " cambio" min = " 0 " max = " 1 " paso = " 0,01 " valor = " 0 " />`

los `< entrada tipo = " número" >` tiene sentido cuando el rango de valores válidos es limitado, por ejemplo, la edad o la altura de una persona. Si el rango es demasiado grande para que los aumentos incrementales tengan sentido (como los códigos postales de EE. UU., Que van desde 00001 a 99999), los `< entrada tipo = " tel " >` podría ser una mejor opción; proporciona el teclado numérico mientras renuncia a la función de interfaz de usuario de ruleta del número.

4.5. Controles deslizantes

Otra forma de elegir un número es usar un **deslizador**. Los ve con bastante frecuencia en sitios como los sitios de compra de viviendas donde desea establecer un precio máximo de propiedad para filtrar. Veamos un ejemplo en vivo para ilustrar esto:

En cuanto al uso, los controles deslizantes son menos precisos que los campos de texto. Por lo tanto, se utilizan para elegir un número cuyo *preciso* el valor no es necesariamente importante.

Se crea un control deslizante con el `< entrada >` con su tipo atributo establecido en el valor rango (

`< entrada tipo = " rango" >`). El control deslizante-pulgar se puede mover mediante el mouse o el tacto, o con las flechas del teclado. Es importante configurar correctamente el control deslizante. Con ese fin, se recomienda encarecidamente que establezca el [min](#) , [max](#) y [paso](#) atributos que establecen los valores mínimo, máximo y de incremento, respectivamente.

Veamos el código detrás del ejemplo anterior, para que pueda ver cómo se hace. En primer lugar, el HTML básico:

```
1. < formar >
2.   < pag > Elija un precio máximo de la casa: </ pag >
3.   < entrada tipo = " rango" nombre = " rango"
4.       min = " 50000 " max = " 500000 " paso = " 100 " valor = " 250000 "
5.       oninput = " número.valor = este.valor " /> < entrada tipo = " número"
6.       nombre = " número"
7.       min = " 50000 " max = " 500000 " paso = " 100 " valor = " 250000 "
8.       oninput = " range.value = this.value " />
9. </ formar >
```

Este ejemplo crea un control deslizante cuyo valor puede oscilar entre 50000 y 500000 , que aumenta / disminuye en 100 a la vez. Le hemos dado un valor predeterminado de 250000 , utilizando la `valor` atributo.

Un problema con los controles deslizantes es que no ofrecen ningún tipo de retroalimentación visual sobre cuál es el valor actual. Por eso hemos incluido un `< entrada tipo = " número" >` elemento para contener el valor actual usando algún código JavaScript (profundizaremos en esto en una unidad futura).

4.6. Selectores de fecha y hora

La recopilación de valores de fecha y hora ha sido tradicionalmente una pesadilla para los desarrolladores web. Para una buena experiencia de usuario, es importante proporcionar una interfaz de usuario de selección de calendario, que permita a los usuarios seleccionar fechas sin necesidad de cambiar de contexto a una aplicación de calendario nativa o potencialmente ingresarlas en diferentes formatos que son difíciles de analizar. Por ejemplo, el último minuto del milenio anterior se puede expresar de muchas formas diferentes: 1999/12/31 , 23:59 , 31/12/99 T11: 59 p.m. , etc.

Los controles de fecha HTML están disponibles para manejar este tipo específico de datos, proporcionando widgets de calendario y haciendo que los datos sean uniformes.

Se crea un control de fecha y hora utilizando el `< entrada >` elemento y un valor apropiado para el tipo atributo, dependiendo de si desea recopilar fechas, horas o ambos. Veamos brevemente los diferentes tipos disponibles:

```
1. < pag > < etiqueta >
2.   Hora de fecha local: < entrada tipo = " datetime-local " nombre = " fecha y hora " /> </ etiqueta > </ pag >
3.
4. < pag > < etiqueta >
5.   Mes: < entrada tipo = " mes " nombre = " mes " >
6. </ etiqueta > </ pag >
7. < pag > < etiqueta >
8.   Hora: < entrada tipo = " hora " nombre = " hora " >
9. </ etiqueta > </ pag >
10. < pag > < etiqueta >
11.   Semana: < entrada tipo = " semana " nombre = " semana " >
12. </ etiqueta > </ pag >
```

Todos los controles de fecha y hora se pueden restringir usando el `min` y `max` atributos, con mayor restricción posible a través de la `paso` atributo (cuyo valor varía según el tipo de entrada):

```
1. < etiqueta >
2.   ¿Cuándo es tu cumpleaños?
```


3. `< entrada tipo = " fecha" nombre = " fecha" min = " 1975-01-01 " max = " 2025-12-31 " paso = " 1 " /> </ etiqueta >`
- 4.

4.7. Control del selector de color

Los colores siempre son un poco difíciles de manejar. Hay muchas formas de expresarlos: valores RGB (decimal o hexadecimal), valores HSL, palabras clave, etc.

Se puede crear fácilmente un control de color usando el `< entrada >` elemento con su tipo atributo establecido en el valor `color` . Por ejemplo:

1. `< etiqueta >`
2. Seleccionar el color: `< entrada tipo = " color" nombre = " color" /> </ etiqueta >`
- 3.

Cuando se admite, hacer clic en un control de color tenderá a mostrar la funcionalidad de selección de color predeterminada del sistema operativo para que pueda hacer su elección. Aquí hay un ejemplo en vivo para que lo pruebes:

4.8. Campo de búsqueda

Los campos de búsqueda están pensados para crear cuadros de búsqueda en páginas y aplicaciones. Este tipo de campo se establece mediante el valor `buscar` . Para el tipo atributo:

1. `< entrada tipo = " buscar" nombre = " buscar" placeholder = " Buscar" necesario />`

La principal diferencia entre un texto campo y un buscar campo es cómo el navegador diseña su apariencia. A menudo, buscar los campos están renderizados con esquinas redondeadas; también a veces muestran un "⊗", Que borra el campo de cualquier valor cuando se hace clic en él. Además, en dispositivos con teclados dinámicos, la tecla Intro del teclado puede leer "buscar" o mostrar un icono de lupa.

Otra característica digna de mención es que los valores de un buscar El campo se puede guardar automáticamente y reutilizado para ofrecer autocompletar en varias páginas del mismo sitio web; esto tiende a ocurrir automáticamente en la mayoría de los navegadores modernos.

4.9. Ejercicio propuesto: tipos de entrada HTML5

Cree una página web para mostrar muestras de todos los elementos de entrada en esta sección: correo electrónico, url, número de teléfono, campo numérico, control de diapositivas, fecha y hora, selector de color y campo de búsqueda. Debe incluir al menos dos ejemplos para cada uno de ellos. Tiene que usar párrafos y etiquetas, y también el atributo "requerido" y todas las restricciones de campo necesarias deben establecerse para todos ellos. Verifique el resultado en su navegador y no olvide incluir todas las etiquetas HTML básicas y validar su código. Finalmente, sube el código a tu dominio y verifica el resultado en tu teléfono móvil.

- Pon todas las etiquetas dentro de un `< formar >` contenedor y use un botón de envío para que pueda verificar que los campos están validados correctamente:

```

1.  < formar >
2.      < pag > < etiqueta >
3.          Correo electrónico principal: < entrada tipo = " correo electrónico" nombre = " email1 " necesario /> </ etiqueta > </ pag >
4.
5.      < pag > < etiqueta >
6.          Email secundario: < entrada tipo = " correo electrónico" nombre = " email2 " necesario /> </ etiqueta > </ pag >
7.
8.      < pag > < etiqueta >
9.          Tu propio sitio web: < entrada tipo = " url " nombre = " sitio web1 " necesario /> </ etiqueta > </ pag >
10.
11.     < pag > < etiqueta >
12.         El sitio web de su escuela: < entrada tipo = " url " nombre = " sitio web2 " necesario /> </ etiqueta > </ pag >
13.
14.     . . .
15.     < pag > < botón > Enviar </ botón > </ pag >
dieciséis </ formar >

```

5. Tu primera forma "real"

La sección le proporciona su primera experiencia al crear un formulario web real, incluido el diseño de un formulario simple, implementarlo utilizando los controles de formulario HTML correctos y otros elementos HTML, y describir cómo se envían los datos a un servidor. Ampliaremos cada uno de estos subtemas con más detalle más adelante.

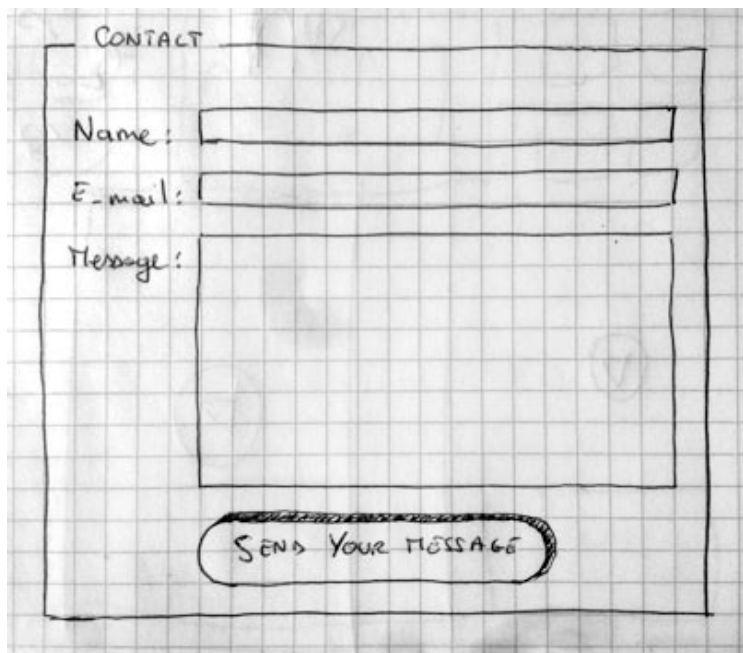
5.1. Diseñando tu formulario

Antes de comenzar a codificar, siempre es mejor dar un paso atrás y tomarse el tiempo para pensar en su formulario. El diseño de una maqueta rápida lo ayudará a definir el conjunto de datos correcto que desea pedirle a su usuario que ingrese. Desde el punto de vista de la experiencia del usuario (UX), es importante recordar que cuanto más grande sea su formulario, más corre el riesgo de frustrar a las personas y perder usuarios. Manténgase simple y manténgase enfocado: solicite solo los datos que absolutamente necesita.

El diseño de formularios es un paso importante al crear un sitio o una aplicación. Está más allá del alcance de esta guía cubrir la experiencia del usuario de los formularios, pero si desea profundizar en ese tema, debe leer los siguientes artículos:

- SmashingMagazine tiene algunos [buenos artículos sobre formularios UX](#) , incluido un antiguo pero aún relevante [Guía extensa de usabilidad de formularios web](#) artículo.
- UXMatters también es un recurso muy reflexivo con buenos consejos de [mejores prácticas básicas](#) a [preocupaciones complejas como formularios de varias páginas](#) .

En esta sección, crearemos un formulario de contacto simple. Hagamos un boceto aproximado:



CONTACT

Name:

E-mail:

Message:

SEND YOUR MESSAGE

Nuestro formulario contendrá tres campos de texto y un botón. Le pedimos al usuario su nombre, su correo electrónico y el mensaje que desea enviar. Al presionar el botón, se enviarán sus datos a un servidor web.

5.2. Implementando tu formulario

Comencemos a crear el HTML para nuestro formulario. Usaremos los siguientes elementos HTML: `< formar >` ,
`< etiqueta >` , `< entrada >` , `< textarea >` , `< botón >` .

5.3. El elemento `<form>`

Todas las formas comienzan con un `< formar >` elemento, así:

```
1. < formar acción = " contact.php " método = " obtener" >  
2.  
3. </ formar >
```

Este elemento define formalmente una forma. Es un elemento contenedor y también admite algunos atributos específicos para configurar la forma en que se comporta. Todos sus atributos son opcionales, pero es una práctica estándar establecer siempre al menos el acción y método atributos:

- los acción El atributo define la ubicación (URL) donde se deben enviar los datos recopilados del formulario cuando se envía.
- los método atributo define con qué método HTTP enviar los datos (normalmente OBTENER o ENVIAR).

Veremos cómo funcionan esos atributos en nuestra sección de envío de datos de formulario más adelante.

5.4. Los elementos `<label>`, `<input>` y `<textarea>`

Nuestro formulario de contacto no es complejo: la parte de entrada de datos contiene tres campos de texto, cada uno con su correspondiente `< etiqueta >` :

- El campo de entrada para el nombre es un campo de texto de una sola línea . El campo de entrada para el correo electrónico es un entrada de tipo email : un campo de texto de una sola línea que acepta solo direcciones de correo electrónico. _____
- El campo de entrada del mensaje es un `< textarea >` : un campo de texto de varias líneas . _____

En términos de código HTML, necesitamos algo como lo siguiente para implementar estos formwidgets:

```
1. < formar acción = " contact.php " método = " OBTENER" >
2.   < pag >
3.     < etiqueta > Nombre: < entrada tipo = " texto" nombre = " nombre" necesario /> </ etiqueta >
4.   </ pag >
5.   < pag >
6.     < etiqueta > Correo electrónico: < entrada tipo = " correo electrónico" nombre = " correo electrónico" necesario /> </ etiqueta >
7.   </ pag >
8.   < pag >
9.     < etiqueta > Mensaje: < textarea nombre = " mensaje" necesario > </ textarea > </ etiqueta >
10.  </ pag >
11. </ formar >
```

Por usabilidad y accesibilidad, incluimos una etiqueta explícita para cada control de formulario. Esto tiene un gran beneficio: asocia la etiqueta con el control de formulario, lo que permite que los usuarios del mouse, el panel táctil y los dispositivos táctiles hagan clic en la etiqueta para activar el control correspondiente, y también proporciona un nombre accesible para que los lectores de pantalla lo lean. a sus usuarios.

Sobre el `< entrada >` elemento, el atributo más importante es el tipo atributo. Este atributo es extremadamente importante porque define la forma en que `< entrada >` El elemento aparece y se comporta:

- En nuestro ejemplo simple, usamos `< entrada tipo = " texto" >` para la primera entrada (el valor predeterminado para este atributo). Representa un campo de texto básico de una sola línea que acepta cualquier tipo de entrada de texto. Para la segunda entrada, usamos `< entrada tipo = " correo electrónico" >` que de ne un campo de texto de una sola línea que solo acepta una dirección de correo electrónico bien formada. Esto convierte un campo de texto básico en una especie de campo "inteligente" que realizará algunas comprobaciones de validación de los datos introducidos por el usuario. También hace que aparezca una distribución de teclado más apropiada para ingresar direcciones de correo electrónico (por ejemplo, con un símbolo @ por defecto) en dispositivos con teclados dinámicos, como teléfonos inteligentes.

Por último, pero no menos importante, tenga en cuenta la sintaxis de `< entrada >` vs. `< textarea > </ textarea >` Esta es una de las rarezas de HTML. los `< entrada >` La etiqueta es un elemento vacío, lo que significa que no necesita una etiqueta de cierre.

`< textarea >` no es un elemento vacío, lo que significa que debe cerrarse con la etiqueta final adecuada.

5.5. El elemento <button>

El marcado de nuestro formulario está casi completo; solo necesitamos agregar un botón para permitir al usuario enviar, o "enviar", sus datos una vez que haya llenado el formulario. Esto se hace usando el < **botón** >

elemento. Solo necesitamos agregar lo siguiente justo encima del cierre </ **formar** > etiqueta:

```
1. < botón tipo = " enviar" > Envíe su mensaje </ botón >
```

Como se explicó en una sección anterior, el < **botón** > elemento también acepta un tipo atributo, con uno de tres valores: enviar , Reiniciar , o botón :

- Un clic en un enviar (el valor predeterminado) envía los datos del formulario a la página web definida por el acción atributo del < **formar** > elemento.
Un clic en un Reiniciar El botón restablece todos los widgets de formulario a su valor predeterminado inmediatamente. Desde el punto de vista
- de UX, esto se considera una mala práctica, por lo que debe evitar usar este tipo de botón a menos que realmente tenga una buena razón para incluir uno.
- Un clic en un botón ¡El botón no hace... nada! Eso suena tonto, pero es increíblemente útil para crear botones personalizados, ya que puede definir la funcionalidad elegida con JavaScript.

5.6. Envío de formdata a su servidor web

La última parte, y quizás la más complicada, es manejar formdata en el lado del servidor. los < **formar** > elemento de nes dónde y cómo enviar los datos gracias al acción y método atributos.

Proporcionamos un nombre a cada control de formulario. Los nombres son importantes tanto del lado del cliente como del servidor; le dicen al navegador qué nombre darle a cada dato y, en el lado del servidor, dejan que el servidor maneje cada dato por su nombre. Los datos del formulario se envían al servidor como pares de nombre / valor.

Para nombrar los datos en un formulario, debe utilizar el nombre atributo en cada widget de formulario que recopilará un dato específico. Veamos nuevamente nuestro formulario de contacto:

```
1. < formar acción = " contact.php " método = " OBTENER" >
2.   < pag >
3.     < etiqueta > Nombre: < entrada tipo = " texto" nombre = " nombre" necesario /> </ etiqueta >
4.   </ pag >
5.   < pag >
```

```

6.      < etiqueta > Correo electrónico: < entrada tipo = " correo electrónico" nombre = " correo electrónico" necesario /> </ etiqueta >
7.      </ pag >
8.      < pag >
9.      < etiqueta > Mensaje: < textarea nombre = " mensaje" necesario > </ textarea > </ etiqueta >
10.     </ pag >
11.     < pag >
12.     < botón tipo = " enviar" > Envíe su mensaje </ botón >
13.     </ pag >
14.     </ formar >

```

En nuestro ejemplo, el formulario enviará 3 datos llamados "nombre", "correo electrónico" y "mensaje". Esa información se enviará a la URL "contact.php" utilizando el método HTTP GET.

En el lado del servidor, el script en la URL "contact.php" recibirá los datos como una lista de tres elementos clave / valor contenidos en la solicitud HTTP. La forma en que este script manejará esos datos depende de usted. Cada lenguaje del lado del servidor (PHP, Python, Ruby, Java, C #, etc.) tiene su propio mecanismo de manejo de datos de formulario. Está más allá del alcance de esta guía profundizar en ese tema para cada idioma, pero le proporcionaremos un ejemplo para que pueda probar sus propios formularios usando PHP.

5.6. 1. Válida de la forma de cl i ent-s ide

Antes de enviar datos al servidor, es importante asegurarse de que se llenen todos los controles de formulario requeridos, en el formato correcto. Se llama **validación de formularios del lado del cliente**, y ayuda a garantizar que los datos enviados coincidan con los requisitos establecidos para los distintos controles de formulario.

La validación del lado del cliente es una verificación inicial y una característica importante de una buena experiencia de usuario; Al capturar datos no válidos en el lado del cliente, el usuario puede corregirlos de inmediato. Si llega al servidor y luego es rechazado, un retraso notable es causado por un viaje de ida y vuelta al servidor y luego de regreso al lado del cliente para decirle al usuario que corrija sus datos.

Si visita cualquier sitio popular con un formulario de registro, notará que brindan comentarios cuando no ingresa sus datos en el formato que esperan. Recibirás mensajes como:

- "Este campo es obligatorio" (no puede dejar este campo en blanco).
- "Ingresa su número de teléfono en el formato xxxxxxxx" (se requiere un formato de datos específico para que se considere válido).

- “Ingresa una dirección de correo electrónico válida” (los datos que ingresó no están en el formato correcto).
- “Su contraseña debe tener entre 8 y 30 caracteres y contener una letra mayúscula, un símbolo y un número” (se requiere un formato de datos muy específico para sus datos).

Se llama **validación de formulario**. Cuando ingrese datos, el navegador y / o el servidor web verificará que los datos estén en el formato correcto y dentro de las restricciones establecidas por la aplicación. La validación realizada en el navegador se llama **lado del cliente** validación, mientras que la validación realizada en el servidor se llama

lado del servidor validación. En este capítulo, nos centraremos en la validación del lado del cliente.

Si la información está formateada correctamente, la aplicación permite que los datos se envíen al servidor y (normalmente) se guarden en una base de datos; si la información no está formateada correctamente, le da al usuario un mensaje de error que explica lo que debe corregirse y le permite volver a intentarlo.

Una de las características más significativas de [Controles de formulario HTML5](#) es la capacidad de validar la mayoría de los datos del usuario. Esto se hace usando atributos de validación en elementos de formulario. Hemos visto muchos de estos anteriormente en la unidad, pero para recapitular:

- [necesario](#) : Especifica si es necesario completar un campo de formulario antes de enviarlo.
- [longitud mínima](#) y [longitud máxima](#) : Especifica la longitud mínima y máxima de los datos textuales (cadenas). Puede restringir la longitud de caracteres de todos los campos de texto creados por `< entrada >` o `< textarea >` utilizando estos atributos. Un campo no es válido si tiene menos caracteres que el longitud mínima valor o más que el longitud máxima valor.
- [min](#) y [max](#) : Especifica los valores mínimo y máximo de los tipos de entrada numérica.
- [tipo](#) : Especifica si los datos deben ser un número, una dirección de correo electrónico o algún otro tipo predeterminado específico.
- [patrón](#) : Especifica a [expresión regular](#) que define un patrón que los datos ingresados deben seguir.

Si los datos ingresados en un campo de formulario siguen todas las reglas especificadas por los atributos anteriores, se considera válido. Si no es así, se considera inválido.

5.6.1.1. Otras lecturas

Queremos que el llenado de formularios web sea lo más fácil posible. Entonces, ¿por qué insistimos en validar nuestros formularios? Hay tres razones principales:

- **Queremos obtener los datos correctos, en el formato correcto.** Nuestras aplicaciones no funcionarán correctamente si los datos de nuestros usuarios se almacenan en un formato incorrecto, son incorrectos o se omiten por completo.
 - **Queremos proteger los datos de nuestros usuarios.** Obligar a nuestros usuarios a ingresar contraseñas seguras facilita la protección de la información de su cuenta.
 - **Queremos protegernos.** Hay muchas formas en las que los usuarios malintencionados pueden hacer un mal uso de formularios no protegidos para dañar la aplicación (consulte [Seguridad del sitio web](#)).
-

Manteniendo esto en mente, validación del lado del cliente *no debe ser considerado* una medida de seguridad exhaustiva! Sus aplicaciones siempre deben realizar comprobaciones de seguridad en los datos enviados por formulario en el *lado del servidor* **también** como del lado del cliente, debido a que la validación del lado del cliente es demasiado fácil de eludir, los usuarios defectuosos aún pueden enviar fácilmente datos incorrectos a su servidor. Leer [Seguridad del sitio web](#) para tener una idea de lo que *podría* suceder; La implementación de la validación del lado del servidor está más allá del alcance de este módulo, pero debe tenerlo en cuenta.

5.6. 2. En el lado del servidor: recuperar los datos

Cualquiera que sea el método HTTP que elija, el servidor recibe una cadena que se analizará para obtener los datos como una lista de pares clave / valor. La forma de acceder a esta lista depende de la plataforma de desarrollo que utilice y de los marcos específicos que pueda utilizar con ella.

[PHP](#) ofrece algunos objetos globales para acceder a los datos. Suponiendo que ha usado el OBTENER , el ejemplo de las siguientes secciones solo toma los datos y los guarda en un archivo. Por supuesto, lo que hagas con los datos depende de ti. Puede mostrarlo, almacenarlo en una base de datos, enviarlo por correo electrónico o procesarlo de alguna otra manera. Usaremos PHP para completar nuestros ejemplos.

5.6. 3. El método GET

los OBTENER El método es el método utilizado por el navegador para pedirle al servidor que envíe un recurso determinado: "Hola servidor, quiero obtener este recurso". En este caso, el navegador envía un cuerpo vacío. Debido a que el cuerpo está vacío, si se envía un formulario utilizando este método, los datos enviados al servidor se adjuntan a la URL.

Teniendo en cuenta nuestro formulario de contacto y teniendo en cuenta que OBTENER Se ha utilizado el método, cuando enviamos el formulario, veremos que los datos aparecen en la URL en la barra de direcciones del navegador. Por ejemplo, si

ingresa "Fernando" como nombre de usuario, " fernando@test.com " como dirección de correo electrónico y "Hola" como mensaje, y presiona el botón enviar, debería ver algo como esto en la dirección bar: " contacto. php ? name = Fernando & email = fernando @ test. com & message = Hola ".

Los datos se adjuntan a la URL como una serie de pares de nombre / valor. Una vez finalizada la dirección web URL, se incluye un signo de interrogación (?) seguido de los pares nombre / valor, cada uno separado por un ampersand (&). En este caso, estamos pasando tres datos al servidor:

- nombre , que tiene un valor de "Fernando"
- correo electrónico , que tiene un valor de " fernando@test.com "
- mensaje , que tiene un valor de "Hola"

5.6.3.1. Ejercicio propuesto: formulario de contacto

Cree una nueva página web con un formulario de contacto, utilizando el código del ejemplo anterior. Debería verse como el de abajo (probablemente no tan bonito). Verifique el resultado en su navegador y valide su código. También intente enviar los datos presionando el botón y verifique la URL dentro de la barra de direcciones. Por último, establezca la longitud mínima y máxima de los campos de texto en los valores que considere adecuados para asegurarse de que los datos de este formulario sean correctos antes de enviarlos al servidor.

- Tenga en cuenta que si presiona el botón enviar, irá a la página "contact.php", que aún no está implementada. En este punto, obtendrá un error, pero verá toda la información en la URL, ya que estamos usando el método GET.

5.6.3.2. Ejercicio propuesto: formulario de contacto completo

Sigamos adelante con un código PHP simple para guardar nuestros datos del formulario de contacto. Cree un archivo "contact.php" con el siguiente código. Sube el formulario y el código php a tu servidor y prueba tu ejemplo completo del formulario de contacto para verificar que los mensajes estén ahora guardados en el servidor. También dile a algunos amigos que prueben la página web y verifiquen que los datos que ingresaron también estén guardados.

```

2. // La variable global $_GET le permite acceder a los datos enviados con el método GET por nombre
3. $ nombre = $_GET ['nombre'];
4. $ correo electrónico = $_GET ['correo electrónico'];
5. $ mensaje = $_GET ['mensaje'];
6.
7. // Ponemos todos los datos en el archivo "messages.csv" en una nueva línea cada vez
8. file_put_contents ( "messages.csv" , "$ nombre; $ correo electrónico; $ mensaje \ n" , FILE_APPEND ) ;
9.
10. // Mostramos un enlace a la página anterior y también al archivo para comprobar los resultados
11. eco "< p> Datos guardados </p> " ;
12. eco "< p> Haga clic en <a href = \" . PS SERVER [HTTP_REFERER]. \"> Aquí </a> volver < /pag > ";
13. eco " < pag > Hacer clic < a href = 'messages.csv' objetivo = '_blanco' > aquí < /una > para ver todos los mensajes < /pag > ";
14. ? >

```

5.6.3.3. Ejercicio propuesto: formulario de saludos

Cree una nueva página web con un formulario similar al siguiente, verifique el resultado en su navegador y valide el código. Presione el botón enviar y eche un vistazo a la barra de direcciones del navegador. Luego ingrese otros datos diferentes a los valores predeterminados, presione el botón Enviar y verifique que la nueva URL contenga la información correcta. Finalmente, cambie el valor predeterminado de ambos campos de texto ("Hola" y "Mamá") para usar otros valores y verifique el resultado nuevamente.

- Tenga en cuenta que si presiona el botón Enviar, irá a la página "greetings.php", que aún no está implementada. En este punto, obtendrá un error, pero verá toda la información en la URL, ya que estamos usando el método GET.

```

1. < formar acción = " greetings.php " método = " OBTENER" >
2.   < pag >
3.     < etiqueta >
4.       ¿Qué saludo quieres decir?: < entrada nombre = " decir" valor = " Hola" necesario
5.     < / etiqueta >
6.   < / pag >
7.   < pag >
8.     < etiqueta >

```

```

9.      ¿A quién quieres decírselo?: < textarea nombre = " a"
necesario > Mamá </ textarea >
10.      </ etiqueta >
11.      </ pag >
12.      < pag >
13.      < botón > Manda mis saludos </ botón >
14.      </ pag >
15.      </ formar >

```

5.6.3.4. Ejercicio propuesto: formulario completo de saludos

Sigamos adelante con un código PHP simple para guardar nuestros datos del formulario de saludos. Cree un archivo "greetings.php" con el siguiente código. Cargue el formulario y el código php en su servidor y pruebe su ejemplo completo del formulario de saludos para verificar que los saludos ahora estén guardados en el servidor. También dile a algunos amigos que prueben la página web y verifiquen que los datos que ingresaron también estén guardados.

- Verás las similitudes del ejemplo anterior (solo hemos cambiado las variables (**PS decir** y **PS a**) y el nombre del archivo donde se guardan los datos ("greetings.csv").

```

1.      < ? php
2.      // La variable global $_GET le permite acceder a los datos enviados con el método GET por nombre
3.      $ decir = $_GET ['decir'] ;
4.      $ a = $_GET ['a'] ;
5.
6.      // Ponemos todos los datos en el archivo "greetings.csv" en una nueva línea cada vez
7.      file_put_contents ( "saludos.csv" , "$ say, $ to \n" , FILE_APPEND ) ;
8.
9.      // Mostramos un enlace a la página anterior y también al archivo para comprobar los resultados
10.     eco "< p> Datos guardados </p> " ;
11.     eco "< p> Haga clic en <a href = \" . PS SERVER ['HTTP_REFERER']. \"> Aquí </a> volver </pag > ";
12.
13.     eco " < pag > Hacer clic < a href = 'saludos.csv' objetivo = '_ blanco' > aquí < /una > para ver todos los mensajes < /pag > ";
13.     ? >

```

5.6.4. El método POST

los ENVIAR El método es un poco diferente. Es el método que utiliza el navegador para hablar con el servidor cuando solicita una respuesta que tiene en cuenta los datos proporcionados en el cuerpo de la solicitud HTTP: "Hola servidor, mira estos datos y envíame un resultado apropiado". Si se envía un formulario con este método, los datos se adjuntan al cuerpo de la solicitud HTTP en lugar de la URL. Es más seguro que el OBTENER método, ya que cuando el formulario se envía utilizando el ENVIAR método, los datos no pueden ser vistos por ninguna otra persona alrededor. Este método se recomienda, por ejemplo, para ser utilizado en formularios donde se envía una contraseña.

Veamos el siguiente ejemplo, que es bastante similar al formulario en el OBTENER sección anterior, pero con el método atributo establecido en ENVIAR y el tipo del cuadro de entrada establecido en "contraseña":

```
1. < formar acción = " login.php " método = " ENVIAR" >
2.   < pag >
3.     < etiqueta > Usuario: < entrada tipo = " texto" nombre = " usuario" necesario /> </ etiqueta >
4.   </ pag >
5.   < pag >
6.     < etiqueta > Contraseña: < entrada tipo = " contraseña" nombre = " contraseña" necesario /> </ etiqueta >
7.   </ pag >
8.   < pag >
9.     < botón > Verificar usuario y contraseña </ botón >
10.  </ pag >
11. </ formar >
```

5.6.4.1. Ejercicio propuesto: formulario de inicio de sesión

Cree una nueva página web con un formulario de inicio de sesión, utilizando el código del ejemplo anterior. Debería verse como el de abajo (probablemente no tan bonito). Verifique el resultado en su navegador y valide su código. También intente enviar los datos presionando el botón y verifique si hay alguna información en la URL. Finalmente, establezca la longitud mínima del campo de texto de usuario en 5 y el máximo a 10, y haga lo mismo con el campo de contraseña.

- Tenga en cuenta que si presiona el botón enviar, irá a la página "login.php", que aún no está implementada. En este punto obtendrá un error, pero no verá ninguna información en la URL, ya que estamos utilizando el método POST.

5.6.4.2. Ejercicio propuesto: formulario de inicio de sesión completo

Sigamos adelante con un código PHP simple para verificar el usuario y la contraseña del formulario de inicio de sesión. Cree un archivo "login.php" con el siguiente código. Cargue el formulario y el código php en su servidor y pruebe su ejemplo completo del formulario de inicio de sesión para verificar el usuario ("admin") y la contraseña ("1234"). También dile a algunos amigos que prueben tu página web. Después de eso, cambie la contraseña del archivo "login.php" y pida a sus amigos que intenten adivinar su nueva contraseña. Debes usar una contraseña muy simple de " <https://en.wikipedia.org/wiki/L> "(De lo contrario, es posible que sus amigos no puedan adivinarlo).

- Verás las similitudes del ejemplo anterior (solo hemos cambiado las variables (**PS usuario** y **PS contraseña**) y hemos utilizado una condición para mostrar una imagen con el pulgar hacia arriba o hacia abajo, dependiendo de si la contraseña es correcta o no.

```

1.  < ? php
2.      // La variable global $_POST le permite acceder a los datos enviados con el método POST por nombre

3.      $ usuario = $_POST ['usuario'] ;
4.      $ contraseña = $_POST ['contraseña'] ;
5.
6.      // Comprueba el usuario y la contraseña
7.      Si ( $ usuario == " administración" && PS contraseña == " 1234 " ) {
8.          eco "< img
src = 'https://raw.githubusercontent.com/twbs/icons/main/icons/hand-thumbs-up.svg' width = '100' /> " ;

9.          eco "< p> ¡Perfecto! :-)</p> <p> Haga clic en <a href = "" . PS SERVER ['HTTP_REFERER']. "">
Aquí </a> volver </pag > ";
10.      }
11.      else {
12.          eco "< img
src = 'https://raw.githubusercontent.com/twbs/icons/main/icons/hand-thumbs-down.svg' ancho = '100' /> " ;

13.          eco "< pag > Usuario invalido o ¡contraseña! : - (</pag > < pag > Hacer clic < a href = "" . $ _ SERVIDOR [' HTTP_REFERER
']. "" > aquí < /una > a tratar de nuevo </pag > ";
14.      }
15.  ? >

```