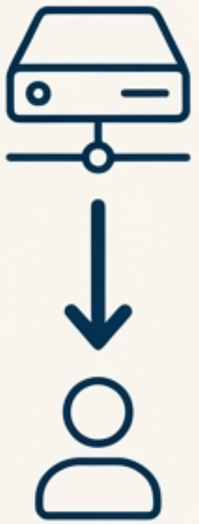
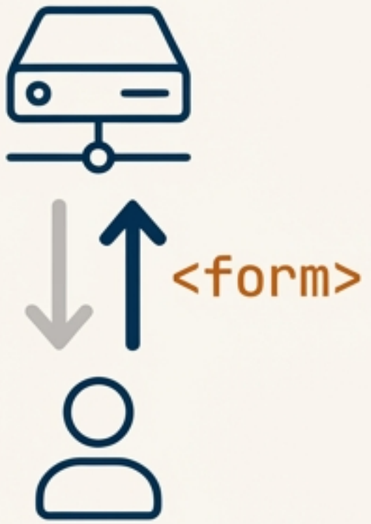
De Páginas a Conversaciones: El Poder de los Formularios HTML

Los formularios son el principal punto de interacción entre un usuario y un sitio web. Permiten la introducción de datos que se envían a un servidor o se usan para actualizar la interfaz.



Páginas Estáticas



Aplicaciones Interactivas

En esta guía, seguiremos el viaje de un constructor. Empezaremos con los cimientos, reuniremos las herramientas y construiremos un formulario funcional, accesible y seguro, entendiendo no solo el 'cómo', sino el 'porqué' de cada pieza.

El Contenedor: Dónde Empieza la Conversación

Todo formulario comienza con el elemento `<form>`. **Define el contenedor y, más importante, a dónde y cómo** se envían los datos.

```
<form action="/ruta/al/servidor.php" method="post">
  <!-- Aquí van los controles del formulario -->
</form>
```

Desglose de Atributos

action: La URL que procesará los datos del formulario. Es el destino de la información.

method: El método HTTP para enviar los datos. Los dos más comunes son `GET` y `POST`. Lo veremos en detalle más adelante.

Nota de Práctica Estándar: Aunque son opcionales, es una práctica estándar establecer siempre los atributos `action` y `method`.

Los Ladrillos Fundamentales: `input` y su Compañero Esencial, `label`

No se puede hablar de un campo de entrada sin hablar de su etiqueta. Son una unidad. El elemento <label> es el más importante para crear formularios accesibles.

Dos Métodos de Asociación

Implícita (Envolviendo el input)

```
<label>Nombre: <input type="text" name="nombre" />
</label>
```

Explícita (con `for` y `id`)

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

Beneficio Adicional (UX)

¡También se puede hacer clic en las etiquetas! Al configurar correctamente las etiquetas, puedes hacer clic o pulsar en la etiqueta para activar el control de formulario correspondiente. Esto es especialmente útil para casillas de verificación y botones de opción, cuya zona sensible puede ser muy pequeña.

La Caja de Herramientas (Parte 1): Recogiendo Texto

1. Texto de una Sola Línea (`<input type="text">)

El control de formulario más básico. Es el valor por defecto si se omite el atributo `type`.

```
<label>Nombre: <input type="text" name="nombre"
  placeholder="Escriba aquí su nombre" required
  minlength="5" maxlength="10"></label>
```

2. Contraseña (`<input type="password">)

Oculta el valor introducido (con puntos o asteriscos) para impedir que otros puedan leerlo. No añade restricciones especiales al texto.

> Esto solo se aplica a nivel de interfaz. A menos que envíes tu formulario en modo seguro (`https://`), se enviará como texto plano. Los navegadores modernos advierten sobre esto.

3. Texto de Múltiples Líneas (`<textarea>`)

Para permitir que los usuarios introduzcan una cantidad considerable de texto, como un comentario.

Diferencia Clave: `<textarea>` no es un elemento vacío. El valor por defecto se define entre sus etiquetas, no con el atributo `value`.

```
<label>Cuéntanos tu historia:
 <textarea name="historia" rows="5"
  cols="30">Era una noche oscura...</
  </textarea></label>
```

La Caja de Herramientas (Parte 2): Ofreciendo Opciones

Estos controles permiten al usuario elegir entre una o más opciones predefinidas. Su comportamiento al enviar el formulario es especial: sus valores se envían solo si están seleccionados.

1. Casillas de Verificación (`<input type="checkbox">)

Permite seleccionar cero o más opciones de un conjunto.


```
<label><input type="checkbox" name="interes"
value="deporte" checked> Deporte</label>
```

2. Botones de Opción (`<input type="radio">)

Permite seleccionar solo una opción de un grupo. Se agrupan compartiendo el mismo valor en el atributo `name`.


```
<label><input type="radio" name="nivel"
value="principiante"> Principiante</label><br>
<label><input type="radio" name="nivel"
value="avanzado" checked> Avanzado</label>
```

3. Menú Desplegable (`<select>)

Crea un menú de opciones representadas por elementos <option>. Útil para listas largas.


```
<label for="mascota">Escoge tu mascota:</label>
<select name="mascota" id="mascota">
  <option value="">--Selecciona una opción--
</option>
  <option value="perro">Perro</option>
  <option value="gato">Gato</option>
</select>
```

Herramientas Especializadas: La Revolución de HTML5

HTML5 añadió nuevos valores para el atributo `type` que imponen restricciones de validación específicas y mejoran la experiencia de usuario, especialmente en dispositivos móviles.

email

Obliga a escribir una dirección de correo válida. `multiple` permite varias direcciones separadas por comas.

url

Valida que el formato incluya un protocolo (ej. `http:`).

tel

No impone restricciones de formato debido a la variedad global, pero a menudo activa un teclado numérico en móviles.

number

Permite solo números. Usa `min`, `max` y `step` para un control preciso.

search

Estilizado como un campo de búsqueda (a veces con una 'X' para limpiar) y puede ofrecer autocompletado basado en búsquedas previas en el sitio.

Controles Visuales Nativos: Más Allá del Texto

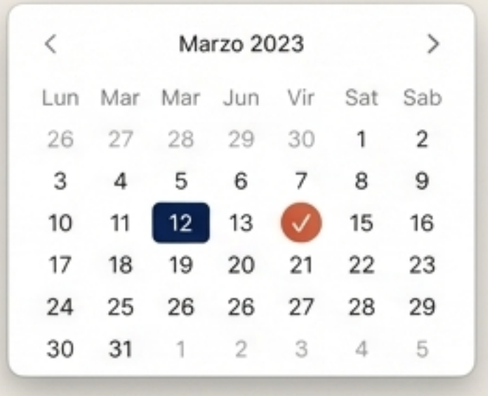
Recopilar fechas, colores o rangos numéricos era una pesadilla. HTML5 proporciona widgets de interfaz de usuario nativos para simplificarlo.

1. Selectores de Fecha y Hora

Proporcionan widgets de calendario para una experiencia de usuario consistente.

Tipos: `date`, `time`, `datetime-local`, `month`, `week`.

```
<label>Tu cumpleaños: <input type="date"
name="fecha" min="1975-01-01">></label>
```



2. Selector de Color

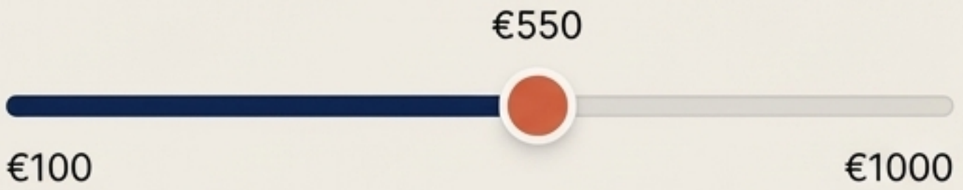
`type="color"` abre el selector de color nativo del sistema operativo.

```
<label>Selecciona un color:
<input type="color" name="color"></label>
```

3. Control Deslizante (Slider)

`type="range"` se usa para seleccionar un número cuyo valor exacto no es necesariamente importante. Es menos preciso que un campo de texto.

Práctica recomendada: Siempre usar con `min`, `max` y `step`. Combínalo con un `<input type="number">` o un elemento `<output>` para mostrar el valor actual.



La Puesta en Marcha: Botones para la Acción

El elemento <button> permite al usuario ejecutar una acción, como enviar o reiniciar el formulario. Su comportamiento se define con el atributo `type`.

`submit` (valor por defecto)

Envía los datos del formulario a la URL definida en el atributo `action`.

`reset`

Restablece todos los controles a sus valores iniciales.

> Desde el punto de vista de la experiencia del usuario, esto se considera una mala práctica. Evítalo a menos que tengas una muy buena razón.

`button`

No tiene comportamiento predeterminado. Es un lienzo en blanco para ser controlado con JavaScript.

<button> vs. <input>

El elemento <button> es más flexible. A diferencia de <input>, no es un elemento vacío y puede contener HTML, como imágenes o texto con formato.

Simple:

```
<input type="submit"
value="Enviar">
```

Enviar

Flexible:

```
<button type="submit"><img
src="icon-send.svg" alt="">
Enviar Mensaje</button>
```

Enviar Mensaje

El Viaje de los Datos: `GET` vs. `POST`

Método `GET` (La Postal)

El navegador pide al servidor un recurso. Los datos del formulario se añaden a la URL como pares `nombre=valor`.

```
contactar.php?nombre=Fernando&correo=test@test.com
```

Cuándo usarlo

Para operaciones no sensibles, como búsquedas o filtros, donde es útil poder marcar la URL como favorita.

Limitaciones

Longitud de URL limitada, datos visibles.

Método `POST` (La Carta Sellada)

El navegador envía datos al servidor en el cuerpo de la petición HTTP, no en la URL.

Cuándo usarlo

Siempre para datos sensibles (contraseñas, datos personales) o cuando los datos son muy largos.

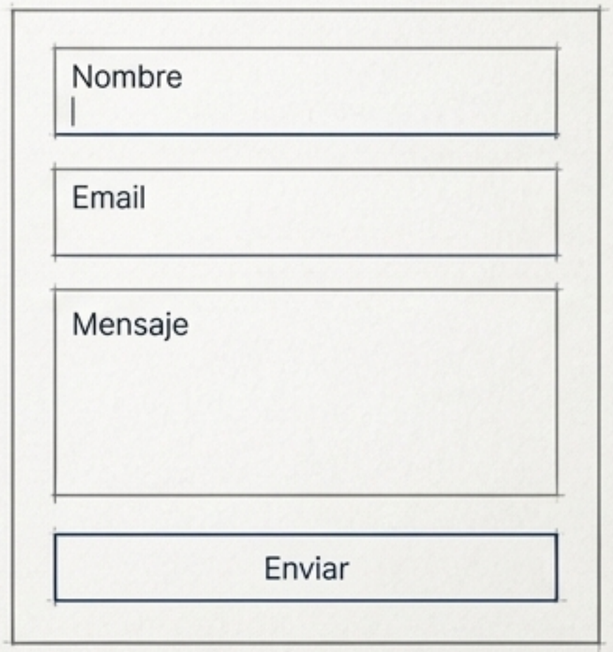
Seguridad

Más seguro que `GET` porque los datos no son visibles en la URL, el historial del navegador o los logs del servidor de la misma manera.

Caso Práctico: Construyendo un Formulario de Contacto

Paso 1: El Diseño (Boceto)

‘Antes de escribir código, es mejor diseñar una maqueta rápida. Ayuda a definir los datos que necesitamos pedir. Recuerda: cuanto más grande es un formulario, más riesgo hay de frustrar al usuario. Pide solo lo necesario.’



Paso 2: El Código Ensamblado

Usaremos `<form>`, `<label>`, `<input type="text">`, `<input type="email">`, `<textarea>` y `<button type="submit">`. Cada control tendrá un `name` para identificar los datos en el servidor.

```
<form action="contactar.php" method="POST">
  <p>
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre" required />
  </p>
  <p>
    <label for="correo">Correo:</label>
    <input type="email" id="correo" name="correo" required />
  </p>
  <p>
    <label for="mensaje">Mensaje:</label>
    <textarea id="mensaje" name="mensaje" rows="5" required></t
  </p>
  <p>
    <button type="submit">Enviar mensaje</button>
  </p>
</form>
```

La Conversación Inteligente: Validación Nativa en el Navegador

¿Por qué validar?

1.  **Obtener datos correctos:** Las aplicaciones no funcionarán bien con datos mal formateados.
2.  **Proteger al usuario:** Forzar contraseñas seguras, por ejemplo.
3.  **Protegerse a uno mismo:** Evitar que usuarios maliciosos dañen la aplicación.

La validación en el lado del cliente es una verificación inicial y una característica importante para una buena UX. Detecta errores al instante, antes de enviar nada al servidor.

Atributos de Validación Nativos

`required`	El campo no puede estar vacío.
`minlength` y `maxlength`	Longitud mínima y máxima para texto.
`min` y `max`	Valor mínimo y máximo para tipos numéricos.
`type`	Valida el formato (ej. `email`, `url`).
`pattern`	Usa una expresión regular para una validación personalizada y compleja.

‘La validación en el lado del cliente no es una medida de seguridad exhaustiva. Es fácil de evitar. Siempre debes realizar comprobaciones de seguridad también en el lado del servidor.’

Feedback Visual Instantáneo con CSS

Cuando un campo de formulario es válido o inválido, el navegador le aplica pseudoclases de CSS especiales. Podemos usarlas para estilizar los campos y guiar al usuario.

Las Pseudoclases en Acción

:valid

Se aplica al elemento cuando su contenido es válido según sus reglas de validación.

```
input:valid { border-color: #003D5B; }
```

:invalid

Se aplica cuando el contenido es inválido.

```
input:invalid { border-color: #D17A22; }
```

El Resultado

Si el usuario intenta enviar un formulario con campos `:invalid`, el navegador bloquea el envío y muestra un mensaje de error, mejorando la usabilidad sin una sola línea de JavaScript.

El Destino Final: Cómo el Servidor Recibe los Datos

El servidor recibe los datos como una lista de pares clave/valor. La 'clave' es el atributo `name` que definimos en nuestro HTML, y el 'valor' es lo que el usuario introdujo.

Ejemplo en PHP

Cada lenguaje de servidor tiene su propio mecanismo. En PHP, se usan variables superglobales como `$_GET` o `$_POST`.

```php
<?php
    // Accedemos a los datos enviados con el método POST
    $nombre = $_POST['nombre'];
    $correo = $_POST['correo'];
    $mensaje = $_POST['mensaje'];

    // Guardamos los datos en un archivo
    $linea = "$nombre,$correo,$mensaje\n";
    file_put_contents("mensajes.csv", $linea, FILE_APPEND);

    // Mostramos un mensaje de éxito
    echo "<p>¡Gracias! Tu mensaje ha sido guardado.</p>";
?>
```

Esto cierra el círculo. El dato introducido por el usuario en el navegador ha sido procesado y almacenado por el servidor, completando la 'conversación'.

Resumen de Atributos Clave y Buenas Prácticas

Atributos Comunes para Controles

`name`: **Esencial.** Identifica el dato para el servidor.

`value`: El valor inicial del control.

`placeholder`: Texto de ayuda que desaparece al escribir.

`disabled`: El control no es usable y su valor no se envía.

`readonly`: El valor no se puede modificar, pero sí se envía.

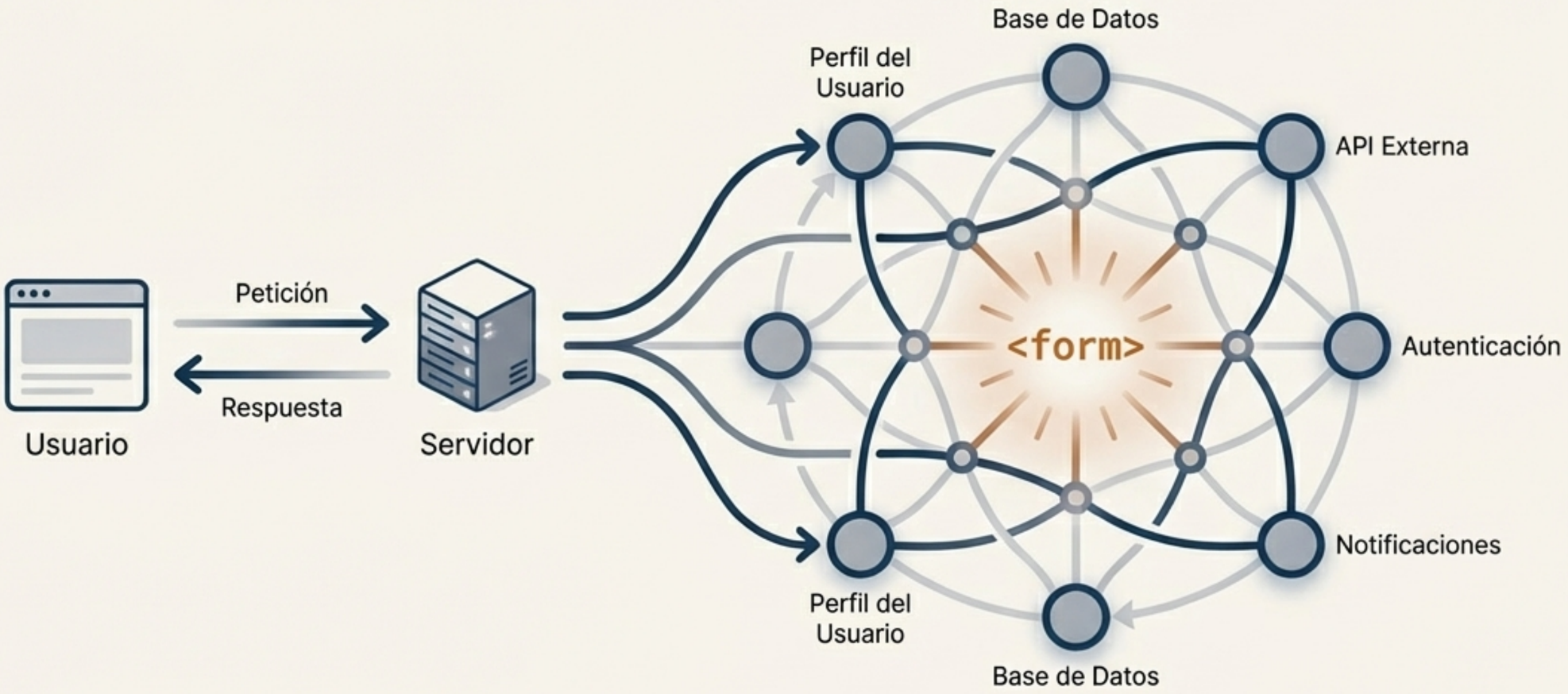
`autofocus`: El control recibe el foco automáticamente al cargar la página.

Checklist de Buenas Prácticas

- [ ] **Accesibilidad primero**: Usa siempre `<label>` y asócialo correctamente.
- [ ] **Elige el `type` correcto**: Aprovecha la validación y UX nativa de HTML5.
- [ ] **Valida en el cliente**: Usa `required`, `pattern`, etc., para guiar al usuario.
- [ ] **Usa `GET` para lo visible, `POST` para lo sensible***: Protege los datos de tus usuarios.
- [ ] **NUNCA confíes en la validación del cliente** : Valida siempre de nuevo en el servidor.

Has Construido el Puente

Dominar los formularios es más que aprender etiquetas. Es entender cómo construir el diálogo entre tus usuarios y tu aplicación. Es el primer paso para crear experiencias web verdaderamente interactivas.



"Ahora tienes la caja de herramientas y los planos. El siguiente paso es construir. Experimenta con cada control, combina sus atributos y crea interfaces que no solo recojan datos, sino que ayuden y guíen a tus usuarios."