

Introducción al Desarrollo web

Curso: Desarrollo Frontend con React

Code Editor



HTML5 Avanzado: Formularios, Tablas y Validaciones

Agenda



1 Tablas

Estructuras para la presentación organizada de datos en filas y columnas

2

Formularios

Elementos interactivos que permiten la entrada de datos por parte del usuario

3

Validaciones

Técnicas para verificar y asegurar la integridad de los datos introducidos

Repaso: Estructura básica de HTML5

Antes de sumergirnos en los elementos avanzados, recordemos la estructura fundamental de un documento HTML5:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Mi página HTML5</title>
</head>
<body>
  <h1>¡Hola mundo!</h1>
  <p>Este es mi primer documento HTML5.</p>
</body>
</html>
```

Elementos estructurales clave:

- `<!DOCTYPE html>`: Declaración del tipo de documento
- `<html>`: Elemento raíz
- `<head>`: Metadatos e información no visible
- `<body>`: Contenido visible

Introducción a tablas en HTML5

Las tablas en HTML5 se utilizan para presentar datos tabulares con una estructura clara:

```
<table>
  <caption>Resultados de Ventas del Primer Trimestre</caption>
  <thead>
    <tr>
      <th>Producto</th>
      <th>Ventas</th>
      <th>Ingresos</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Portátiles</td>
      <td>120</td>
      <td>72.000€</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <th>Total</th>
      <td>370</td>
      <td>159.500€</td>
    </tr>
  </tfoot>
</table>
```



Recuerda: Las tablas deben usarse exclusivamente para datos tabulares, no para maquetar páginas web.

Elementos estructurales de tablas



<table>

Elemento raíz que define la tabla



<caption>

Título o descripción de la tabla



<thead>

Sección de encabezado, contiene filas de cabecera



<tbody>

Sección principal, contiene las filas de datos



<tfoot>

Sección de pie, útil para totales o resúmenes



<tr>

Define una fila de la tabla



<td>

Celda estándar para datos



<th>

Celda de encabezado, con semántica especial

Esta estructura semántica es esencial para la accesibilidad y el correcto procesamiento de los datos.

Extensión de celdas y columnas

```
<table>
  <thead>
    <tr>
      <th>Nombre</th>
      <th>Contacto</th>
      <th colspan="2">Horarios</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td rowspan="2">Servicio técnico</td>
      <td>soporte@ejemplo.com</td>
      <td>Lunes - Viernes</td>
      <td>8:00 - 20:00</td>
    </tr>
    <tr>
      <td>urgencias@ejemplo.com</td>
      <td>Sábado - Domingo</td>
      <td>9:00 - 14:00</td>
    </tr>
  </tbody>
</table>
```

The screenshot shows a table with 4 columns and 6 rows. Row 1 has 3 columns and 1 colspan="2" cell. Row 2 has 4 cells. Row 3 has 4 cells. Row 4 has 4 cells. Row 5 has 4 cells. Row 6 has 4 cells. Cells are styled with different backgrounds (white, blue, light blue) and borders.

X <	HTML Table		
X			
■ ●	● ○	■ ○	■ ■
—	—	—	—
—	—	—	—
—	—	—	—

Atributos para combinar celdas:

- `colspan="n"`: Extiende una celda a lo largo de n columnas
- `rowspan="n"`: Extiende una celda a lo largo de n filas

Accesibilidad en tablas

Las tablas accesibles son esenciales para usuarios con lectores de pantalla:

<caption>

Proporciona un título descriptivo que explica el propósito de la tabla.

```
<caption>Horario de clases del primer semestre</caption>
```

scope

Indica si una celda de encabezado se aplica a una fila o columna.

```
<th scope="col">Nombre</th>
<th scope="row">Ana</th>
```

headers e id

Asocia celdas de datos con sus encabezados correspondientes.

```
<th id="nombre">Nombre</th>
<td headers="nombre">Juan Pérez</td>
```

summary

Aunque obsoleto en HTML5, proporciona información adicional sobre la estructura de la tabla.

```
<table summary="Datos de ventas mensuales por región para el año fiscal 2023">
```

Importante: Las tablas bien estructuradas son cruciales para los usuarios que navegan con tecnologías de asistencia.

Introducción a Formularios HTML5

Elemento `<form>` y sus atributos principales

El elemento `<form>` actúa como contenedor para todos los controles interactivos de entrada de datos:

```
<form action="/procesar-datos" method="post" enctype="multipart/form-data" autocomplete="on">
  <!-- Aquí se añadirían los elementos de control del formulario, como inputs, textareas, etc. -->
  <label for="username">Nombre de usuario:</label><br>
  <input type="text" id="username" name="username"><br>
  <button type="submit">Enviar</button>
</form>
```

action

URL que procesará el formulario cuando se envíe

method

Método HTTP utilizado para enviar datos (get o post)

enctype

Tipo de codificación para el envío (importante para archivos)

autocomplete

Activa/desactiva el autocompletado del navegador (on/off)

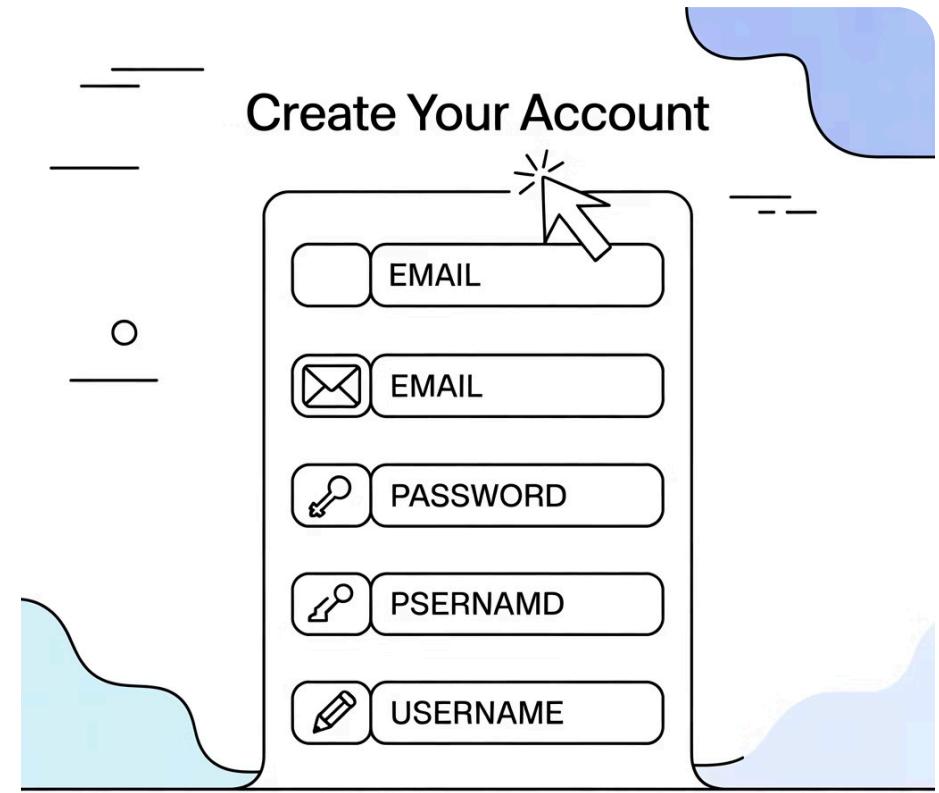
Etiquetado de controles: <label>

El elemento <label> es fundamental para la accesibilidad y usabilidad:

```
<!-- Método 1: Usando el atributo "for" -->  
<label for="nombreUsuario">Nombre:</label>  
<input type="text" id="nombreUsuario" name="nombreUsuario">  
  
<!-- Método 2: Anidando el input dentro del label -->  
<label>  
  Apellido:  
  <input type="text" name="apellidoUsuario">  
</label>
```

Beneficios del etiquetado correcto:

- Mejora la accesibilidad para lectores de pantalla
- Aumenta el área clicable (al hacer clic en la etiqueta)
- Proporciona contexto claro al usuario



[TERMS OF SERVICE](#) | [PRIVACY POLICY](#)

Consejo: Siempre usa el elemento <label> para cada control de formulario, incluso cuando el diseño visual no lo requiera explícitamente.

Campos de entrada: <input>

El elemento <input> es el más versátil para la entrada de datos, con múltiples tipos disponibles en HTML5:



Texto

`type="text"`

Entrada básica de texto de una sola línea



Email

`type="email"`

Dirección de correo electrónico



Contraseña

`type="password"`

Campo que oculta los caracteres



Teléfono

`type="tel"`

Número telefónico



Fecha

`type="date"`

Selector de fecha



Número

`type="number"`

Entrada numérica con controles

Ventaja: Cada tipo proporciona validación intrínseca y experiencia de usuario optimizada según el dispositivo.

Tipos avanzados de <input>

```
<label for="colorInput">Elige un color:</label>
<input type="color" id="colorInput" name="color" value="#ff0000">

<label for="rangeInput">Valor (0-100):</label>
<input type="range" id="rangeInput" name="points" min="0" max="100" value="50">

<label for="urlInput">Tu sitio web:</label>
<input type="url" id="urlInput" name="website" placeholder="https://ejemplo.com">

<label for="searchInput">Buscar:</label>
<input type="search" id="searchInput" name="query" placeholder="Escribe aquí para buscar">

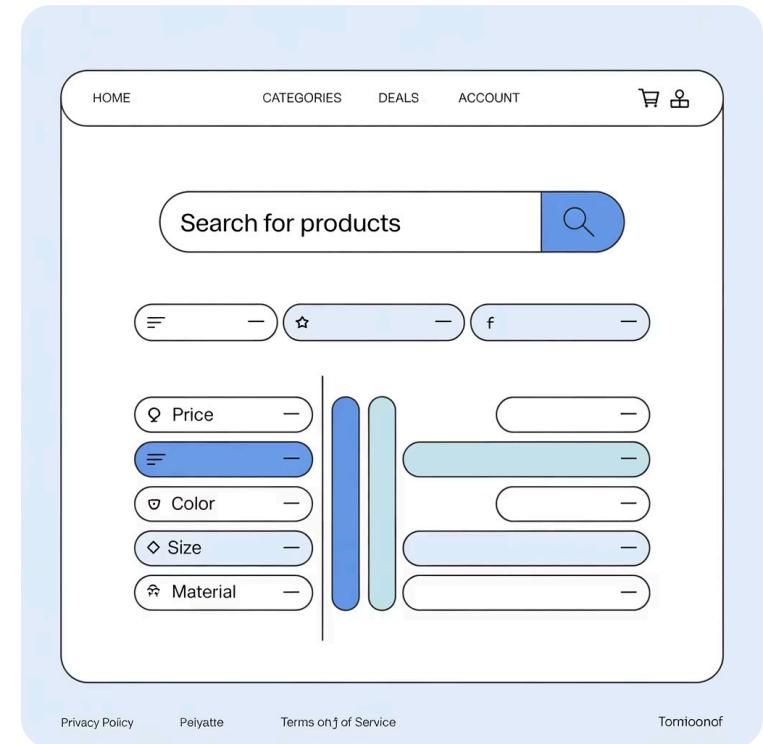
<label for="timeInput">Hora de la reunión:</label>
<input type="time" id="timeInput" name="meetingTime" value="14:30">

<label for="datetimeLocalInput">Fecha y Hora del evento:</label>
<input type="datetime-local" id="datetimeLocalInput" name="eventDateTime">

<label for="weekInput">Semana del año:</label>
<input type="week" id="weekInput" name="weekOfYear">

<label for="monthInput">Mes y año:</label>
<input type="month" id="monthInput" name="monthAndYear">

<label for="fileInput">Subir archivo:</label>
<input type="file" id="fileInput" name="document" accept=".pdf,.doc,.docx">
```



Estos tipos especializados ofrecen interfaces intuitivas para diferentes tipos de datos, mejorando la experiencia del usuario sin necesidad de JavaScript.

Importante: La compatibilidad con navegadores antiguos debe considerarse. Siempre implementa fallbacks apropiados.

Áreas de texto y selección múltiple

<textarea>: Texto multilínea

```
<label for="comment">Comentario:</label>
<textarea id="comment" name="userComment" rows="5" cols="40"
placeholder="Escribe tu comentario aquí...">Texto predeterminado (opcional)
</textarea>
```

Atributos importantes:

- `rows`: Número de líneas visibles
- `cols`: Ancho visible en caracteres
- `resize`: Controlar redimensionamiento (CSS)

<select>: Listas desplegables

```
<label for="province">Provincia:</label>
<select id="province" name="province">
<option value="">-- Selecciona una opción --</option>
<option value="madrid">Madrid</option>
<option value="barcelona">Barcelona</option>
<option value="valencia">Valencia</option>
<optgroup label="Andalucía">
<option value="sevilla">Sevilla</option>
<option value="malaga">Málaga</option>
</optgroup>
</select>
```

El atributo `multiple` permite selección múltiple:

```
<label for="languages">Idiomas:</label>
<select id="languages" name="languages" multiple size="3">
<option value="es">Español</option>
<option value="en">Inglés</option>
<option value="fr">Francés</option>
</select>
```

Botones y controles de selección

Botones

```
<!-- Botón de envío (para formularios) -->  
<button type="submit">Enviar</button>  
  
<!-- Botón de reinicio (para formularios) -->  
<button type="reset">Limpiar</button>  
  
<!-- Botón genérico (para scripts JavaScript) -->  
<button type="button">Más información</button>
```

[Enviar](#) [Limpiar](#) [Más información](#)

Casillas y radios

```
<!-- Grupo de casillas de verificación (checkboxes) -->  
<label>Intereses:</label>  
<input type="checkbox" id="interest_prog" name="interests" value="programming">  
<label for="interest_prog">Programación</label>  
<input type="checkbox" id="interest_webdesign" name="interests" value="web_design">  
<label for="interest_webdesign">Diseño web</label>
```



```
<!-- Grupo de botones de radio (radio buttons) -->  
<label>Nivel de experiencia:</label>  
<input type="radio" id="exp_beginner" name="experience" value="beginner">  
<label for="exp_beginner">Principiante</label>  
<input type="radio" id="exp_intermediate" name="experience" value="intermediate">  
<label for="exp_intermediate">Intermedio</label>  
<input type="radio" id="exp_advanced" name="experience" value="advanced">  
<label for="exp_advanced">Avanzado</label>
```

Intereses: Programación Diseño web

Nivel de experiencia: Principiante Intermedio Avanzado

Agrupación y estructuración de formularios

Los elementos `<fieldset>` y `<legend>` permiten agrupar controles relacionados:

```
<form>
  <fieldset>
    <legend>Datos personales</legend>
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre" required>
    <br><br>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
  </fieldset>

  <fieldset>
    <legend>Datos de envío</legend>>
    <label for="direccion">Dirección:</label>
    <input type="text" id="direccion" name="direccion" required>
    <br><br>
    <label for="postal_code">Código postal:</label>
    <input type="text" id="postal_code" name="postal_code" required>
  </fieldset>

  <br>
  <button type="submit">Enviar pedido</button>
</form>
```

Ejemplo de formulario estructurado con **Datos personales** (Nombre, Email) y **Datos de envío** (Dirección, Código postal), incluyendo un botón de envío.

Beneficio: Mejora significativamente la organización visual y la accesibilidad del formulario.

Validación básica con atributos HTML5

HTML5 proporciona atributos para validación nativa del navegador sin JavaScript:

required

Hace que un campo sea obligatorio

```
<input type="text" name="nombre" required>
```

pattern

Valida mediante expresión regular

```
<input type="text" name="codigo" pattern="[A-Z]{3}-[0-9]{3}"  
title="Formato: ABC-123">
```

min / max

Establece límites para valores numéricos

```
<input type="number" name="edad" min="18" max="65">
```

minlength / maxlength

Controla la longitud del texto

```
<input type="text" name="usuario" minlength="3"  
maxlength="20">
```

Estas validaciones se ejecutan automáticamente al enviar el formulario y muestran mensajes de error nativos del navegador.

Validación avanzada con pattern

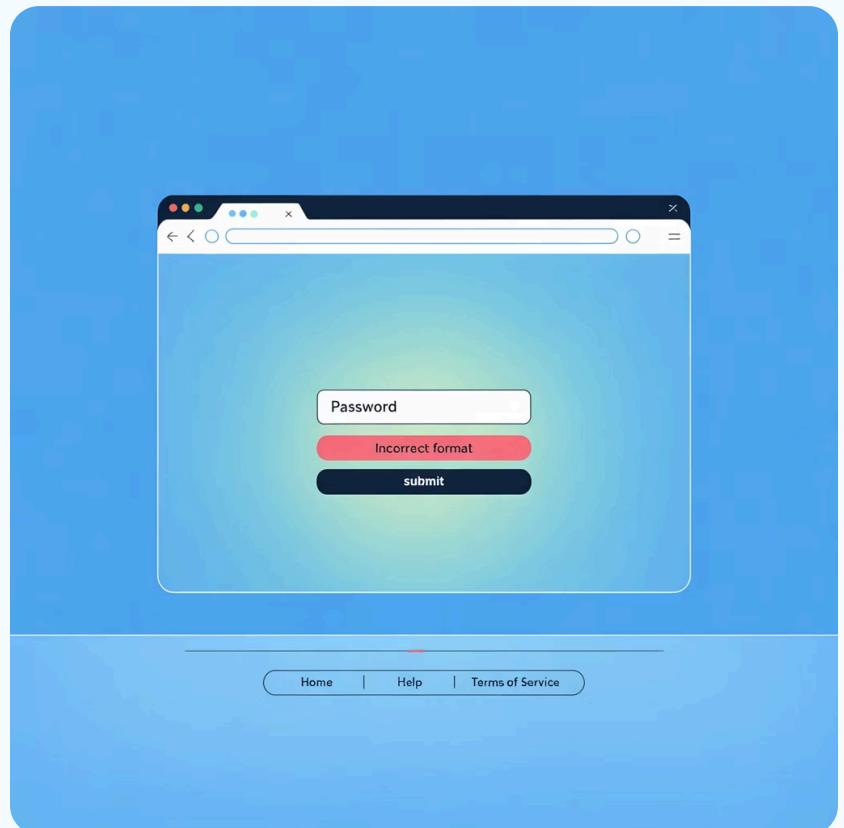
El atributo `pattern` permite crear validaciones complejas mediante expresiones regulares:

```
<!-- Contraseña (mín. 8 caracteres, al menos 1 mayúscula, 1 minúscula, 1 número, 1 símbolo) -->
<label for="password">Contraseña:</label>
<input type="password" id="password" name="password"
       pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&*]).{8,}"
       title="Debe contener al menos un número, una mayúscula, una minúscula, un símbolo (@#$%^&*) y 8 o más
             caracteres" required>

<!-- Código Postal (5 dígitos) -->
<label for="cp">Código Postal:</label>
<input type="text" id="cp" name="cp" pattern="^\d{5}$"
       title="Debe ser un código postal de 5 dígitos" required>

<!-- Teléfono (9 dígitos, opcionalmente con espacios o guiones) -->
<label for="phone">Teléfono:</label>
<input type="tel" id="phone" name="phone"
       pattern="^(+34|0034)?[\s.-]?( \d{3}[\s.-]?){2}\d{3}$"
       title="Debe ser un número de teléfono válido (9 dígitos)" required>

<!-- DNI (8 dígitos y 1 letra) -->
<label for="dni">DNI:</label>
<input type="text" id="dni" name="dni" pattern="^\d{8}[A-Za-z]$"
       title="Debe ser un DNI válido (8 dígitos y 1 letra)" required>
```



El atributo `title` se muestra como tooltip y como mensaje de error cuando no se cumple el patrón.

Personalización de la validación

Anular la validación nativa

Podemos desactivar la validación con novalidate:

```
<form action="/submit" method="post" novalidate>
  <label for="username">Usuario:</label>
  <input type="text" id="username" name="username" required>
  <button type="submit">Enviar</button>
</form>
```

Guardar borrador

Útil cuando queremos implementar nuestra propia validación con JavaScript o cuando necesitamos permitir envíos parciales.

Mensajes de error personalizados

CSS permite personalizar los mensajes de error:

```
/* Estilo para campos inválidos */
input:invalid {
  border: 2px solid red;
  background-color: #ffdddd;
}

/* Estilo para el mensaje que aparece cuando el campo es inválido */
/* Nota: Este método de mostrar mensajes de error con ::after no es estándar para la
validación nativa de HTML5. */
/* Generalmente, los mensajes personalizados se manejan con JavaScript y elementos HTML
dedicados para el error. */
input:invalid::after {
  content: "Por favor, introduzca un valor válido.";
  display: block; /* Muestra el contenido en una nueva línea */
  color: red;
  font-size: 0.9em;
  margin-top: 5px;
}

/* Estilo para campos válidos */
input:valid {
  border: 2px solid green;
  background-color: #e6ffe6;
}
```

Atributos adicionales para la experiencia de usuario

placeholder

```
<input type="text" placeholder="Introduce tu nombre">
```

Texto de ejemplo que desaparece al empezar a escribir

autocomplete

```
<input type="email" name="email" autocomplete="email">
```

Sugerencias basadas en entradas anteriores

autofocus

```
<input type="text" autofocus>
```

Enfoca automáticamente este campo al cargar

list

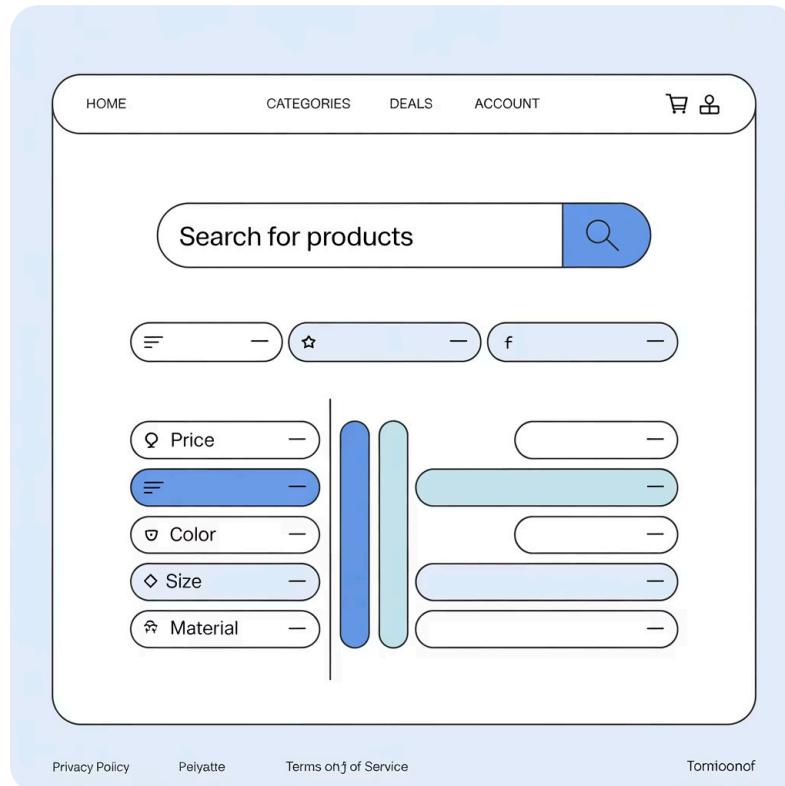
```
<input list="frutas">
<datalist id="frutas">
  <option value="Manzana">
  <option value="Banana">
  <option value="Cereza">
</datalist>
```

Proporciona sugerencias predefinidas

Consejo: Estos atributos mejoran la usabilidad sin necesidad de JavaScript y son especialmente útiles en dispositivos móviles.

Formularios de búsqueda y filtrado

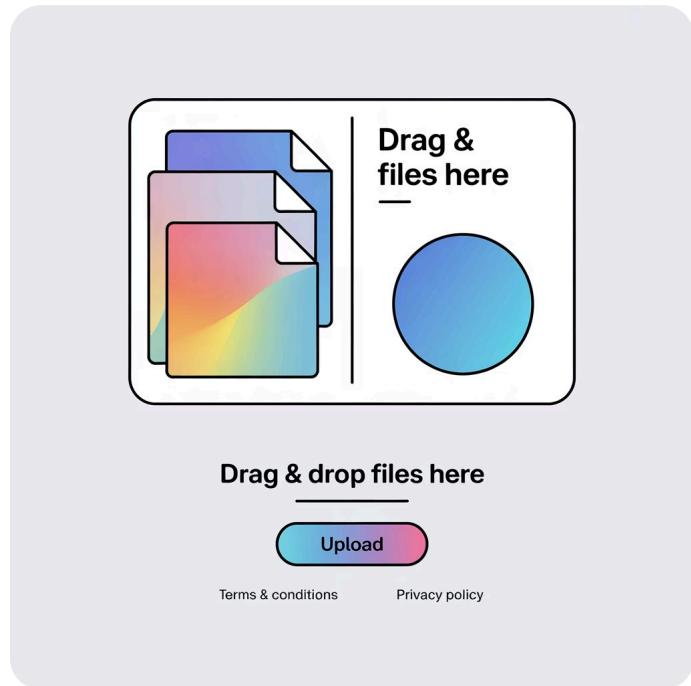
Este ejemplo demuestra una estructura simple para un formulario de búsqueda básico, ideal para capturar una consulta de texto y enviar una petición.



```
<form method="GET" action="/search">
  <label for="searchQuery">Buscar:</label>
  <input type="text" id="searchQuery" name="q" placeholder="Buscar productos...">
  <br><br>
  <button type="submit">Buscar</button>
</form>
```

Este ejemplo utiliza el método GET para que los parámetros de búsqueda sean visibles en la URL y los usuarios puedan guardar o compartir los resultados.

Carga y manejo de archivos



```
<form action="/upload" method="post" enctype="multipart/form-data">
  <label for="singleFile">Subir un documento:</label><br>
  <input type="file" id="singleFile" name="singleFile" accept=".pdf,.doc,.docx,.jpg,.jpeg,.png">
<br><br>

  <label for="multipleFiles">Seleccionar múltiples archivos:</label><br>
  <input type="file" id="multipleFiles" name="multipleFiles[]" multiple><br><br>

  <button type="submit">Subir archivos</button>
</form>
```

Aspectos importantes para la carga de archivos:

- El atributo `enctype="multipart/form-data"` es **obligatorio**
- El atributo `accept` mejora la usabilidad pero no es una validación segura
- El atributo `multiple` permite seleccionar varios archivos
- La validación de tamaño y tipo real debe hacerse en el servidor

Ejercicio práctico: Sitio web de cafetería mejorado

En el ejercicio anterior, creamos la estructura básica HTML5 para el sitio web de una cafetería.

Ese proyecto ya contaba con las secciones principales: un encabezado con navegación, una sección principal con información sobre la cafetería y un pie de página. Ahora vamos a AMPLIAR ese sitio web añadiendo dos nuevas secciones importantes:

1. Sección "Contáctanos" con formulario:

- Formulario de contacto completo con fieldset para agrupar información
- Campos para: nombre, email, teléfono, tipo de consulta (select), mensaje
- Validación HTML5 apropiada para cada campo
- Checkbox para aceptar política de privacidad
- Botones de envío y limpiar

2. Menú con tablas:

- Tabla estructurada para mostrar el menú de la cafetería
- Categorías: Bebidas calientes, Bebidas frías, Postres, Snacks
- Columnas: Producto, Descripción, Precio
- Uso de thead, tbody, caption para estructura semántica
- Aplicar atributos de accesibilidad (scope, headers)

Buenas prácticas para formularios



Accesibilidad

- Usar siempre elementos `<label>` asociados a los controles
- Agrupar campos relacionados con `<fieldset>` y `<legend>`
- Proporcionar mensajes de error claros y específicos
- Asegurar navegación por teclado (orden de tabulación lógico)



Adaptación móvil

- Usar controles suficientemente grandes (mín. 44x44px para tocar)
- Aprovechar tipos de input específicos (`tel`, `email`, etc.)
- Limitar el número de campos al mínimo necesario
- Diseño de una columna para dispositivos pequeños



Seguridad

- Evitar exponer datos sensibles en URLs (usar POST)
- Implementar CSRF tokens en el servidor
- Validar siempre en el servidor, no confiar solo en la validación del cliente
- Usar HTTPS para todos los formularios



Experiencia de usuario

- Mostrar el progreso en formularios largos
- Conservar datos ya introducidos en caso de error
- Ordenar campos de forma lógica y coherente
- Proporcionar ayuda contextual (tooltips, ejemplos)

¿Hasta dónde puede llegar HTML sin JavaScript?

HTML5 ofrece muchas capacidades nativas que antes requerían JavaScript:



Lo que HTML5 puede hacer solo

- Validación básica de campos (obligatorios, formatos, rangos)
- Interfaces específicas para tipos de datos (fechas, colores, etc.)
- Autocompletado y sugerencias
- Mensajes de error nativos
- Estructura semántica clara

Limitaciones sin JavaScript

- Validaciones complejas interdependientes
- Campos condicionales o dinámicos
- Actualización en tiempo real
- Envío asíncrono (AJAX)
- Mensajes de error personalizados
- Animaciones y efectos

Enfoque recomendado

- Usar HTML5 como base sólida (progressive enhancement)
- Asegurar que el formulario funcione sin JavaScript
- Añadir JavaScript para mejorar experiencia y validación
- Siempre validar en el servidor por seguridad

Consejo: Comienza con HTML semántico y válido, luego mejora progresivamente con CSS y JavaScript.

Recursos y herramientas adicionales

Documentación y referencias

- MDN Web Docs: Guía completa sobre formularios HTML
- W3C HTML5 Specification: Estándar oficial
- Can I Use: Compatibilidad con navegadores

Herramientas de validación

- W3C Markup Validation Service
- HTML5 Validator
- WAVE Web Accessibility Evaluation Tool
- Lighthouse (Chrome DevTools)

Frameworks y bibliotecas complementarias

- Bootstrap Forms: Estilos predefinidos
- Formspree: Procesamiento de formularios sin backend
- HTML5 Boilerplate: Plantillas iniciales
- PureCSS: Estilos ligeros para formularios



Resumen y conclusiones



Formularios potentes

HTML5 proporciona herramientas nativas para crear formularios funcionales, accesibles y con validación integrada.

Recuerda que HTML5 debe ser la base sólida sobre la que construir, siguiendo principios de mejora progresiva y diseño centrado en el usuario.

¡Gracias por la atención prestada!

¿Preguntas?



Tablas estructuradas

Las tablas HTML modernas ofrecen una estructura semántica rica para presentar datos de manera accesible y organizada.



Validación eficiente

La validación nativa elimina código JavaScript innecesario y proporciona una capa inicial de verificación de datos.

#EDCOUNIANDES

<https://educacioncontinua.uniandes.edu.co/>

Contacto: educacion.continua@uniandes.edu.co

© - Derechos Reservados: La presente obra, y en general todos sus contenidos, se encuentran protegidos por las normas internacionales y nacionales vigentes sobre propiedad Intelectual, por lo tanto su utilización parcial o total, reproducción, comunicación pública, transformación, distribución, alquiler, préstamo público e importación, total o parcial, en todo o en parte, en formato impreso o digital y en cualquier formato conocido o por conocer, se encuentran prohibidos, y solo serán lícitos en la medida en que se cuente con la autorización previa y expresa por escrito de la Universidad de los Andes.