

El lenguaje HTML

Historia

La primera versión de HTML se publicó en 1989. Tim Berners-Lee, conocido popularmente como “el padre de Internet” la presentó en su día junto con el concepto de hipertexto, algo parecido a lo que hoy conocemos como enlaces web. La primera documentación oficial, sin embargo, se hizo esperar hasta 1991, siendo estandarizado poco después.

En 1998 hubo un gran salto cualitativo en HTML con la publicación de la cuarta versión del lenguaje: HTML4. En esta versión se implementan por primera vez las hojas de estilo en cascada (CSS) y se regulariza el uso de scripts. Poco después, desde principios del siglo XXI estuvo muy en auge XHTML, cuyo objetivo era adaptar HTML 4.01 al lenguaje XML, incluyendo entre otras cosas extensiones sobre las versiones previas de HTML. El manejo de errores y la sintaxis eran claramente más estrictos que en HTML estándar. Sin embargo, a pesar de que algunas etiquetas de XHTML han perdurado en el tiempo, su desarrollo ha sido abandonado en favor de HTML5.



El estándar actual, HTML5, fue publicado por primera vez en 2014. Fue desarrollado en paralelo a XHTML y es el que se ha ido imponiendo en la World Wide Web. Incluye numerosos elementos nuevos sobre versiones anteriores de HTML, eliminando otros que se consideran obsoletos. También incluye bastantes etiquetas que no existían para dar soporte a elementos multimedia o recoger tipos de datos nuevos que han ido surgiendo desde la publicación de versiones previas del lenguaje. Otra gran innovación es la inclusión de las etiquetas semánticas, que no presentan contenido en pantalla pero sirven para organizar y estructurar el contenido de las páginas HTML a nivel interno.

Organización de un documento HTML

Los archivos HTML son archivos de texto plano que se pueden crear o modificar con cualquier editor de texto convencional, si bien existen muchos entornos especializados que facilitan el trabajo de los desarrolladores y nos autocompletan etiquetas, entre otras ventajas.

Un fichero HTML se estructura como un conjunto de etiquetas encerradas entre ángulos: `<...>`. Algunas etiquetas requieren ser abiertas y cerradas: `<...>` contenido `</...>`, mientras que otras no requieren cierre. Con el conjunto de etiquetas que forman el documento HTML se crea una estructura de tipo árbol, con el elemento `<html>` como raíz. En esta estructura podemos incluir numerosos niveles de organización, con elementos que definan secciones generales del documento y elementos que representen secciones más específicas o contenido a visualizar.

El lenguaje HTML dispone de varias etiquetas estructurales que deberíamos utilizar en todos los documentos que creemos, las cuales definen la organización interna de elementos importantes en la página. De entre ellas destacaremos las siguientes:

- `<!DOCTYPE html>` . Siempre aparece al principio del archivo, y sirve para indicar que lo que viene a continuación es una página HTML.
- `<html>... </html>` Delimita el código HTML. Es la raíz de toda la estructura de la página, por lo que todos los elementos deberían aparecer entre estas etiquetas. Se puede y de hecho es recomendable especificar el idioma de la página. Por ejemplo, para el idioma español escribiremos `<html lang="es">`
- `<head>... </head>` Entre estas etiquetas definiremos la información necesaria para configurar la página web, como el título, el conjunto de caracteres, los metadatos y los enlaces a las hojas de estilo. Nada de lo que se especifique en la etiqueta head se muestra en pantalla.
- `<body>...</body>` Aquí se define todo el contenido de la página web que se va a visualizar en pantalla. Todo el contenido visible debe estar incluido dentro de la etiqueta body. Dentro del cuerpo de la página podremos incluir tanto elementos de línea como elementos de bloque (definiremos lo que son más adelante).

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5      <title>Title goes here</title>
6    </head>
7    <body>
8
9    </body>
10   </html>
```

Formateo de texto en HTML5

A diferencia de XHTML, HTML5 es muy permisivo con errores de formato. En HTML5 el navegador siempre intentará mostrar todo lo que pueda de las páginas que se le presentan. A pesar de esta flexibilidad, es recomendable seguir unas sencillas pautas para aumentar la legibilidad del código generado:

- Usar solamente letras minúsculas para las etiquetas.
- Usar comillas dobles para los atributos dentro de las etiquetas.
- Cerrar siempre todas las etiquetas, incluso las que no requieren el cierre.
- Siempre indentar el código que escribamos.

El formato genérico de las etiquetas en HTML es el siguiente:

```
<nombre_etiqueta atributo_1="valor" atributo_2="valor" ...
atributo_n="valor">
    elementos afectados por la etiqueta
</nombre_etiqueta>
```

En HTML también podemos poner comentarios, a título informativo, que no se mostrarán en pantalla. Sirven para documentar nuestro código y tienen el siguiente formato:

```
<!-- Comentario -->
```

Cabecera de un archivo HTML. Etiqueta <head>

En esta etiqueta se incluyen la información y los recursos necesarios para generar la página correctamente. Algunos de los elementos que pueden aparecer en esta sección son:

- <title>...</title> Muestra el título de la página web en la ventana del navegador.
- <base> Define la URL que utilizará el navegador como referencia sobre la cual se establecerá la URL absoluta a partir de las URL relativas que se especifiquen en el código. Si aparece esta etiqueta solamente puede ser una vez y siempre dentro de la etiqueta <head>. Así, por ejemplo, si en nuestra página escribimos <base href="https://dominio.com"> y más adelante en el código tenemos algo como esa URL de la imagen se traducirá automáticamente en https://dominio.com/fotos/imagen.jpg
- <meta> Representa los metadatos de un documento como la descripción, las palabras clave, el tipo de codificación de caracteres, ... Ejemplos:
 - <meta charset="UTF-8"> Caracteres Unicode, que son más que suficientes para nuestras necesidades.
 - <meta http-equiv="content-type" content="text/html; charset=utf-8"> Otra forma de especificar lo mismo que en el ejemplo anterior.
 - <meta charset="ISO-8859-1"> Caracteres del alfabeto latino. Menos estándar que Unicode.
 - <meta name="description" content="Descripción de la página"> Descripción general en lenguaje natural del contenido, para buscadores.
 - <meta name="author" content="Mi Nombre"> Identifica al autor de la web.
 - <meta name="keywords" content="k1,k2,k3..."> Palabras clave que serán utilizadas principalmente por los motores de búsqueda para posicionar nuestra web.
 - <meta name="viewport" content="width=device-width, initial-scale=1.0"> Es un metadato muy

importante y debemos incluirlo siempre. Da instrucciones al navegador sobre cómo presentar las dimensiones de la página y el escalado de la misma. Si no lo especificamos no podremos tener páginas que se adapten a diferentes tamaños de dispositivos (responsive web).

- `<meta name="generator" content="Nombre de Programa">` Si la página ha sido generada automáticamente por algún software, es habitual que se añada este metadato para indicarlo.
- `<script>...</script>` Se utiliza para insertar código JavaScript en la página. Aunque se puede incluir dentro de la sección `<head>`, es más recomendable ponerlo dentro del cuerpo de la página web, justo al final, antes de `</body>`, para que se cargue la página completa antes de acceder y descargar las líneas de JavaScript. Esto es particularmente útil si estamos accediendo a repositorios externos.
- `<style>...</style>` Con estas etiquetas podemos definir estilos para el archivo HTML que estamos creando. Es muy mala idea usar esta etiqueta en los archivos HTML; crear archivos CSS aparte con todas las especificaciones de estilo hace que todos los elementos de estilo estén juntos en un mismo archivo, se puedan compartir entre páginas diferentes y permitan la consistencia de la página web, así como un mantenimiento menos complicado.
- `<link>` Permite agregar recursos externos a la página. Su uso más habitual es enlazar hojas de estilo necesarias para mostrar adecuadamente el contenido de la página. Un ejemplo de esto sería: `<link rel="stylesheet" href=".css/archivo.css">`. Esta etiqueta también se puede usar para especificar el ícono que mostrará la página en la ventana del navegador o si la agregamos a favoritos.
Ejemplos:
 - `<link rel="icon" type="image/png" href="/images/icon.png">` Archivo de imagen formato png.
 - `<link rel="icon" type="image/x-icon" href="/images/icon.ico">` Archivo de ícono de Windows.
 - `<link rel="icon" type="image/png" href="/images/icon.png" sizes="32x32">` Aquí hemos especificado también el tamaño en píxeles del ícono. El tamaño más estándar es 16x16, aunque también se usa 32x32.

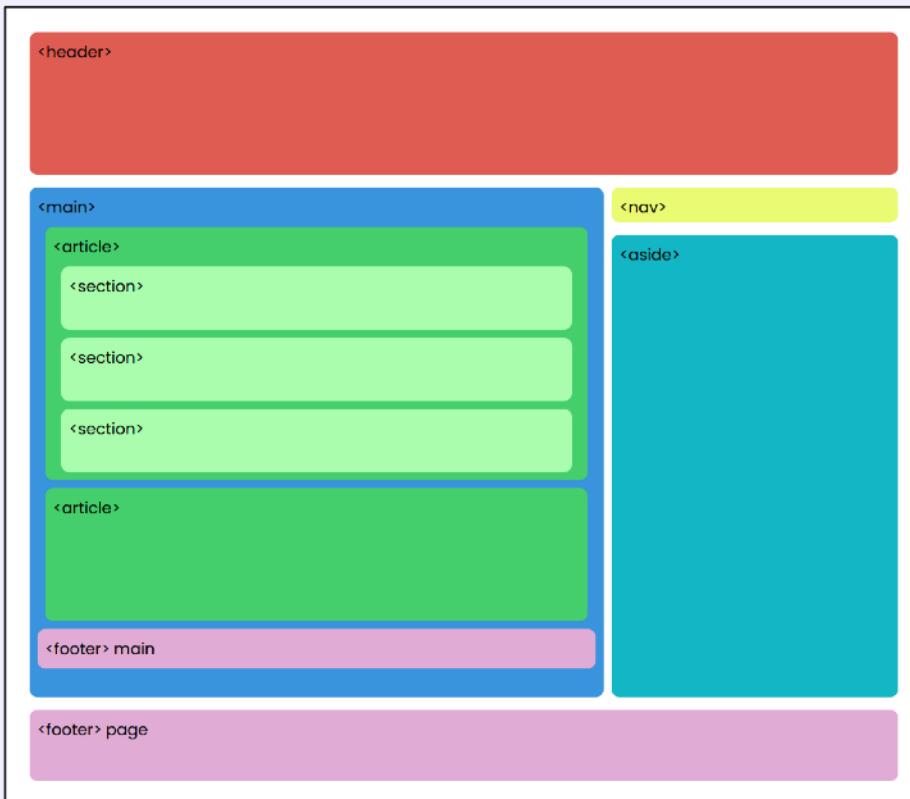
Cuerpo de un archivo HTML. Etiqueta <body>

Dentro de la etiqueta <body> incluiremos todo el contenido visible de la página web. Antiguamente todo el contenido solía estructurarse con tablas utilizando la etiqueta <table>, pero con CSS se consiguen resultados mucho más potentes con menos esfuerzo. Posteriormente se usaron etiquetas separadoras como <div> para estructurar el contenido. Sin embargo, en HTML5 se recomienda usar etiquetas que informen del contenido que se está definiendo, aunque no se muestren en pantalla. Estas etiquetas son las denominadas etiquetas semánticas, y también son utilizadas por los motores de búsqueda para acceder al contenido de las páginas.

Elementos estructurales del cuerpo

- <div>...</div> División genérica. Etiqueta no semántica.
- <main>...</main> Contenido principal de la página.
- <nav>...</nav> Elementos de navegación, menús, etc.
- <header>...</header> Encabezado de la página.
- <footer>...</footer> Pie de página.
- <section>...</section> Subdivisión dentro de un elemento más grande.
- <article>...</article> Para artículos con contenido relevante o informativo.
- <aside>...</aside> Barras laterales, contenido menos importante que el principal.

Aunque cada diseñador puede escoger una distribución personalizada para sus páginas es importante utilizar estas etiquetas semánticas. El motivo es que los "web crawlers" de los motores de búsqueda darán prioridad a las páginas que las tengan. Además, el código con etiquetas semánticas resulta más legible para los desarrolladores. En cualquier caso, siempre deberemos tener presentes todos los principios de diseño que ya conocemos.



HTML5 es muy flexible, pero aun así deberíamos declarar los elementos estructurales en el mismo orden en el que queramos que se vayan a mostrar en pantalla. Mediante CSS podríamos manipular ese orden, pero es más sencillo declararlo correctamente desde el principio.

Atributos globales en elementos estructurales

Estos valores suelen usarse en las hojas de estilo de las páginas web. Algunos elementos se pueden usar muchas veces en un mismo documento (en especial los más genéricos como <div>), de modo que podemos incluir identificadores únicos para los elementos como atributos globales. Son los siguientes:

- **id**: Asigna un identificador único, que no debería repetirse, a un elemento.
- **class**: Asigna un identificador común a uno o varios elementos que compartirán características similares.

Podemos incluir `id` y `class` a la vez en la misma etiqueta, y también podemos poner más de una clase en un elemento siempre que las separemos con espacios. Podemos usar tanto `id` como `class` en los archivos CSS para identificar elementos cuyo estilo deseamos modificar. También podemos acceder a los elementos en código JavaScript por su clase o su identificador, si bien es mucho más habitual utilizar el identificador para este último propósito.

Ejemplos:

```
<div id="identificador"> ... </div>  
<div class="clase1 clase2"> ... </div>
```

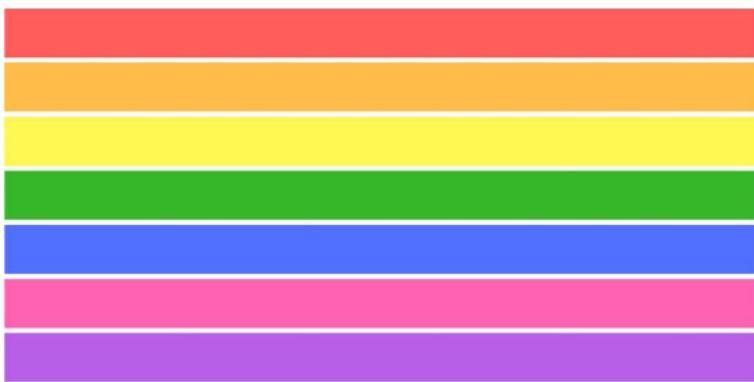
```
1 <HTML>  
2   <input class="is-valid has-icon" id="productID" name="product" value="roses" />  
3   <script>  
4     var product = productID.value;  
5     console.log(product); //roses  
6   </script>  
7 </HTML>
```

Contenido visualizable en un archivo HTML

Dentro de las etiquetas que presentan contenido visible en pantalla podemos distinguir dos tipos fundamentales:

- Elementos de línea (inline). No necesitan comenzar en una nueva línea para mostrarse. Ocupan el espacio estrictamente necesario para su visualización completa.
- Elementos de bloque (block). Siempre comienzan una línea nueva en el contenido y ocupan todo el ancho de la pantalla para mostrarse.

BLOCK-LEVEL ELEMENTS:



INLINE ELEMENTS:



Algunos elementos de línea: `<a>`, `<abbr>`, `<bdo>`, `
`, `<button>`, ``, ``, `<input>`, `<label>`, `<map>`, `<object>`, `<output>`, `<q>`, `<script>`, `<select>`, ``, ``, `<sub>`, `<sup>`, `<textarea>`, ``,...

Algunos elementos de bloque: `<div>`, `<header>`, `<main>`, `<nav>`, `<section>`, `<article>`, `<footer>`, `<aside>`, `<blockquote>`, `<form>`, `<fieldset>`, `<figure>`, `<hr>`, `<p>`, `<table>`, `<td>`, `<tr>`, `<th>`, `<figcaption>`, `<dl>`, `<dr>`, `<dt>`, ``, ``, ``, `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`,...

Elementos de texto en el contenido

- `<h1>...</h1>`, `<h2>...</h2>`, ... `<h6>...</h6>` Representan títulos y subtítulos, incluyendo hasta 6 subniveles. El título principal suele llevar la etiqueta `<h1>`, mientras que los subtítulos sucesivos llevarán etiquetas según su nivel. Por defecto los navegadores otorgan un estilo con un tamaño de letra mayor a estos títulos.
- `<p>...</p>`. Para mostrar párrafos de texto. Dejan un espacio en blanco al final del párrafo, porque `<p>` es un elemento de bloque.

- `<pre>...</pre>` Texto con un formato predefinido, como un poema o fragmentos de código, donde se respetarán los saltos de línea y espacios que pongamos.
- `...`. Elemento para marcar de forma no visible un texto, palabra o frase. Se suele utilizar dentro de las hojas de estilo para destacar alguna parte. Es un elemento de línea.
- `
` Inserta un salto de línea.
- `<wbr>` Define puntos en palabras largas donde se pueden romper para mostrarlas en el navegador si no caben. Una palabra muy larga sin esta etiqueta podría estropear el formato o partirse incorrectamente. Esta ruptura de palabras no dibuja guiones en pantalla; una alternativa para que se muestren es usar el carácter "soft hyphen" (`­` en HTML) en su lugar, que mostrará guiones en las zonas acotadas cuando se pueda. El mismo efecto que `<wbr>` puede conseguirse con la propiedad CSS `word-break`.
 - `palabralarga<wbr>quepartimos<wbr>paramostrarla.`
 - `palabralarga­quepartimos­paramostrarla.`
- `...` Enfatizar texto. Por defecto hace que se muestre en cursiva.
- `...` Texto importante. Por defecto hace que se muestre en negrita.
- `<u>...</u>` Texto con errores ortográficos. Por defecto se muestra subrayado. Esta etiqueta no debería utilizarse porque hace que el texto se confunda con los enlaces, que también se muestran subrayados por defecto.
- `<mark>...</mark>` Texto resaltado. Por defecto se muestra con un fondo de color distinto.
- `_{...}`, `^{...}` Texto en subíndice o en superíndice.
- `<abbr>...</abbr>` Abreviaturas.

Enlaces

Hipertexto: Conectar documentos o páginas entre sí a través de enlaces o hipervínculos. La etiqueta `<a>...` es la que se utiliza para crear enlaces.

El texto que se mostrará en pantalla referente al enlace se pone entre las etiquetas de apertura y cierre. Además, la etiqueta `<a>` puede tener estos atributos:

- `href="Ubicación"` Indica a dónde conduce el enlace. Puede ser una ubicación absoluta, una ubicación relativa o incluso una ubicación interna de la propia página (que se acceden mediante su atributo `id` con `#id`).
- `download`. Descarga el elemento vinculado en lugar de visualizarlo en el navegador.
- `target`. Indica cómo se abre el enlace. Ejemplos:
 - `target="_blank"`. Abre en una pestaña nueva
 - `target="_self"`. Comportamiento por defecto. Abre el enlace en la pestaña actual.
- `type`. Especifica el tipo de contenido al que se está enlazando. No es obligatorio y es exclusivamente informativo.
- `rel`. Explica la relación entre la página que enlaza y la que es enlazada. Ejemplos:
 - `nofollow`. El enlace no será seguido por los "web crawlers" de los motores de búsqueda.
 - `next / prev`. Anterior o siguiente documento en un conjunto.
 - `external`. Documento ajeno a la web en la que nos encontramos.
 - `author / license / bookmark`. Datos sobre derechos, etc.

También podemos crear enlaces a correos electrónicos y teléfonos con los valores `mailto:` y `tel:` respectivamente (dentro del campo `href` del enlace). Sin embargo es poco frecuente encontrar ese tipo de enlace.

Inserción de imágenes

Hay varias etiquetas diferentes que permiten incluir imágenes en un documento HTML. A la más tradicional `` se le han ido uniendo más recientemente `<picture>` y `<figure>`, aparecidas con HTML5.

La etiqueta `` inserta una imagen en la página. Es un elemento de línea y requiere del atributo `src` para especificar la ubicación absoluta o relativa de la imagen. Se recomienda incluir el atributo `alt`, que define un texto alternativo por si la imagen no pudiera mostrarse correctamente. También son interesantes los atributos `width` y `height`, que definen anchura y altura explícitas para la imagen en pantalla (por defecto se tomarán las dimensiones del archivo de imagen). Es interesante señalar que las etiquetas `width` y `height` no toman en consideración el tamaño original de la imagen, por lo que si especificamos tamaños muy diferentes al tamaño real de la imagen podrían producirse distorsiones en lo que se presente en pantalla.

En la etiqueta `` se puede usar el atributo `srcset` para mostrar imágenes diferentes en según qué situaciones, pero se prefiere usar `srcset` únicamente dentro de la etiqueta `<picture>`. En `` también podemos especificar el atributo `usemap`, si se ha definido un mapeado de la imagen con áreas interactivas, las cuales pueden vincularse con otras páginas.

Ejemplo de uso de la etiqueta ``:

```

```

Mapeado de imágenes:

```


<map name="mapa1">

    <area shape="rect" coords="0,0,100,100"
        href="pag2.html" alt="Rectángulo">

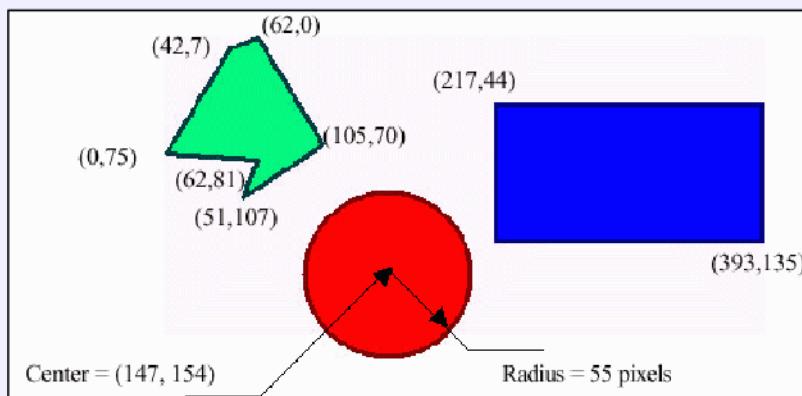
    <area shape="circle" coords="300,300,20"
        href="pag3.html" alt="Círculo">

    <area shape="poly" coords="x1,y1,x2,y2,x3,y3,..."
        href="pag4.html" alt="Polígono">

</map>
```

Para definir las coordenadas de las formas de un mapa de imágenes se sigue el siguiente criterio:

- Para los rectángulos se especifican dos coordenadas: primero la superior izquierda y después la inferior derecha.
- Para los círculos se especifican tres coordenadas: la coordenada (x,y) del centro y el tamaño en píxeles del radio del círculo
- En el caso de polígonos irregulares se definen todas las coordenadas que forman el polígono, en orden.



En HTML5 se ha añadido la etiqueta `<picture>` para aportar más flexibilidad en el manejo de imágenes. Esta etiqueta puede usarse fácilmente en diseños adaptables (responsive) cambiando la imagen concreta que se visualiza en función de la resolución actual de la pantalla. Para lograr este objetivo se incluirán varios elementos `<source>` dentro de `<picture>`; en cada uno de ellos el atributo `media` indicará en qué condiciones se visualizará cada uno de esos elementos `<source>`. La imagen concreta que se va a mostrar en cada caso se define en el atributo `srcset`. Es muy importante poner una etiqueta `` al final de `<picture>`, para el caso de navegadores antiguos que no sean compatibles o como imagen por defecto si ninguno de los elementos `<source>` es aplicable. El uso del elemento `<picture>` puede reducir el ancho de banda y el tiempo de carga de las páginas web cargando imágenes más pequeñas en dispositivos de menos resolución.

Ejemplo de uso de <picture>:

```
<picture>
    <source media="(min-width:700px)" srcset="foto1.jpg">
    <source media="(min-width=400px)" srcset="foto2.jpg">
    
</picture>
```

El elemento <picture> también puede ser útil si empleamos formatos de imagen no soportados por todos los navegadores. Si ponemos varias líneas de <source>, el navegador nos presentará en pantalla la imagen correspondiente a la primera línea que haya sabido interpretar correctamente.

Por último debemos comentar la etiqueta <figure>, que se emplea para definir elementos gráficos como ilustraciones, diagramas o fotografías descriptivas. Dentro de <figure> podemos utilizar la etiqueta <figcaption> para añadir un texto explicativo que陪伴e a la figura. En la etiqueta <figure> se pueden utilizar elementos o <picture> indistintamente.

Ejemplos de uso de <figure>:

```
<figure>
    
    <figcaption>Descripción de la foto</figcaption>
</figure>
```

```
<figure>
    <picture>
        <source media="(min-width:820px)" srcset="fotogrande.jpg">
        <source media="(min-width:6000px)" srcset="fotomediana.jpg">
        
    </picture>
    <figcaption>Figura de prueba</figcaption>
</figure>
```

Listas de elementos en HTML

En HTML tenemos disponibles varias opciones para presentar el contenido con el formato de una lista de ítems. Podemos tener listas ordenadas (numeradas), listas no ordenadas (viñetas) o listas de términos y descripciones.

HTML can have Unordered lists, Ordered lists, or Description lists:

Unordered List

- The first item
- The second item
- The third item
- The fourth item

Ordered List

1. The first item
2. The second item
3. The third item
4. The fourth item

Description List

- | | |
|------------------------|---------------------|
| The first item | Description of item |
| The second item | Description of item |

Listas numeradas:

```
<ol>
    <li> Elemento 1 </li>
    <li> Elemento 2 </li>
    ...
</ol>
```

Estas listas se pueden personalizar con los atributos `reversed` (que muestra los números en orden inverso) `start` (que determina desde qué número comienza la lista) o `type` (que determina qué tipo de indicador se va a utilizar para los números). Algunos ejemplos de esto último:

- 1: Números
- a: Letras minúsculas
- A: Letras mayúsculas
- i: Números romanos en minúsculas
- I: Números romanos en mayúsculas

Ejemplo de lista ordenada, con letras mayúsculas en orden inverso:

```
<ol type="A" reversed>
    <li>Elemento</li>
    <li>Elemento</li>
    <li>Elemento</li>
</ol>
```

Listas sin numerar:

```
<ul>
    <li> Elemento 1 </li>
    <li> Elemento 2 </li>
    ...
</ul>
```

Para cambiar el tipo de viñeta por defecto podemos hacerlo con el atributo `list-style-type` en CSS.

Las listas de términos y descripciones son similares a las listas no numeradas, pero en este caso presentan una lista de términos que se complementan con un texto más largo, el cual explica o describe el término. Es típico en glosarios o definiciones enciclopédicas. Se usa de la siguiente manera:

```
<dt> Término 1 </dt>
    <dd> Definición del término 1 </dd>
<dt> Término 2 </dt>
    <dd> Definición del término 2 </dd>
    ...

```

Algunas etiquetas HTML no están específicamente diseñadas para la definición de listas, pero pueden utilizarse para ello. Es el caso de `<blockquote>` y `<details>`.

La etiqueta `<blockquote>` representa un bloque de texto, que suele ser una cita o un extracto de algún otro texto o documento. Técnicamente no es muy diferente a `<p>`, pero incluye márgenes a los lados, por lo que también puede usarse como una herramienta para presentar listas de valores. Usar `<blockquote>` repetidamente generaría listas no ordenadas sin viñetas.

Ejemplo:

```
<blockquote> Elemento 1 </blockquote>
<blockquote> Elemento 2 </blockquote>
...

```

También puede usarse la etiqueta `<details>` para definir listas. Esta etiqueta crea elementos que revelan información adicional cuando el usuario hace clic sobre él, desplegando su contenido. Dicho contenido puede incluir cualquier etiqueta común de HTML, como `<p>` o `<blockquote>` entre muchas otras. Así podremos usarlo para mostrar una especie de lista desplegable. Eso sí, esta etiqueta requiere el uso de un elemento `<summary>` para definir el texto del cuál "colgará" toda la lista que se vaya a desplegar.

Ejemplo:

```
<details>
  <summary> Título de la lista </summary>
  <p>Elemento 1 </p>
  <p>Elemento 2 </p>
  <p>Elemento 3 </p>
  ...
</details>
```

Tablas en HTML

Las tablas han sido históricamente un método muy habitual para organizar visualmente la información que se presenta en una página web. De hecho

también era frecuente utilizar tablas como soporte para estructurar todo el contenido visual de la página antes de que las hojas de estilo simplificasen esa tarea.

En la actualidad las tablas han caído un poco en desuso porque la tendencia es incluir las tablas como figuras (así pueden ser adaptables fácilmente y su manipulación es más simple). Sin embargo es conveniente conocer la sintaxis porque puede ser útil para futuros proyectos.

- `<table>...</table>` Define una tabla.
- `<tr>...</tr>` Define la fila de una tabla.
- `<td>...</td>` Define una celda dentro de una tabla.
- `<th>...</th>` Define la cabecera de una tabla.
- `<caption>...</caption>` Define el título de una tabla, que se presenta fuera de la misma.

El navegador interpreta el código que introducimos de forma secuencial, por lo que es importante definir la tabla fila por fila, detallando el contenido en celdas de cada una de esas filas.

- Para combinar casillas de una tabla tenemos que utilizar los atributos `rowspan` y `colspan`. Con `rowspan="n"` definiremos una combinación fusionando n filas, mientras que con `colspan="n"` permite la fusión de n columnas. Si queremos poner un pie bajo la tabla podemos poner la etiqueta `<tfoot>`, aunque su contenido no se diferenciará visualmente del resto de la tabla.

Dentro de la etiqueta `<table>`, si ponemos el atributo `border="0"` se mostrará la tabla sin bordes (es el comportamiento por defecto), mientras que si ponemos `border="1"` se presentará una tabla con un borde rodeando cada celda.

Ejemplo: Tabla 2x2 sin fusionar celdas:

```
<table>
  <tr>
    <td> Celda 1 </td>
    <td> Celda 2 </td>
  </tr>
  <tr>
    <td> Celda 3 </td>
    <td> Celda 4 </td>
  </tr>
</table>
```

Celda 1	Celda 2
Celda 3	Celda 4

Ejemplo: tabla con filas y columnas combinadas:

```
<table>
  <tr>
    <td colspan="2"> A </td>
    <td> B </td>
  </tr>
  <tr>
    <td rowspan="2"> C </td>
    <td> E </td>
    <td> F </td>
  </tr>
  <tr>
    <td> Y </td>
    <td> Z </td>
  </tr>
</table>
```

A		B
C	E	F
	Y	Z

Otras etiquetas HTML

- `...` Elemento contenedor, pero a diferencia de los que hemos visto antes (`<div>` o las etiquetas semánticas) es un elemento de línea y no de bloque. Su uso más frecuente es el formateo de partes muy pequeñas de un texto utilizando CSS.
- `<hr>` Muestra una línea horizontal. Se usa para separar textos u otros elementos de la página. No requiere etiqueta de cierre.
- ` ... ` Muestra texto “tachado”, con una línea horizontal que cruza el texto por la mitad.
- `<bdo>... </bdo>` Permite mostrar texto en orden inverso. Si ponemos `<bdo dir="rtl">` el texto que aparezca a continuación se mostrará de derecha a izquierda. La opción por defecto (de izquierda a derecha) es `dir="ltr"`
- `<button> ... </button>` Sirve para crear botones, cuyo uso y función vendrá asociado a la hoja de estilo o estará definido dentro de un script, por poner un par de ejemplos. Dentro de un botón podemos incluir muchos elementos, aparte del texto. Es posible añadir imágenes, líneas o más elementos HTML dentro de `<button>`. Para que el botón tenga sentido es necesario programar su funcionamiento dentro del código, habitualmente usando lenguajes como JavaScript. Algunos atributos de los botones HTML son:
 - `name`: Nombre del botón.
 - `type`: Tipo del botón. Tiene más sentido dentro de formularios (`button, reset, submit`).
 - `value`: Valor del botón, que será el que se envíe en el caso de formularios o scripts.
 - `disabled`: Si aparece, significa que el botón está deshabilitado y no se puede pulsar.

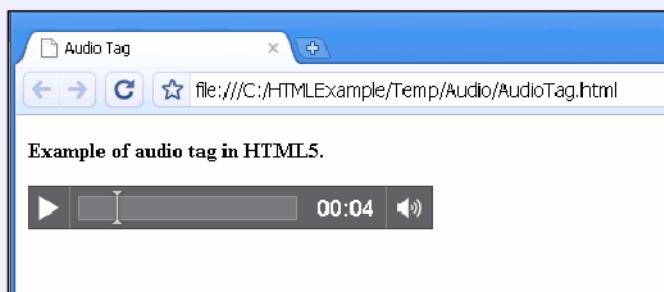
Elementos multimedia en HTML

Ya conocemos el elemento `<picture>` para introducir imágenes en HTML. Desde HTML5 se han añadido etiquetas de un formato muy similar para poder incluir audio y vídeo en una página web.

Para introducir sonido en una página web utilizaremos la etiqueta `<audio>`. Podemos insertar archivos de audio en formato mp3, wav u ogg. Su uso es análogo al de la etiqueta `<picture>`, pudiendo especificar varias fuentes para el audio.

Ejemplo:

```
<audio controls autoplay loop>  
    <source src="audio.ogg" type="audio/ogg">  
    <source src="audio.mp3" type="audio/mpeg">  
    Lo siento, tu navegador no soporta este contenido.  
</audio>
```



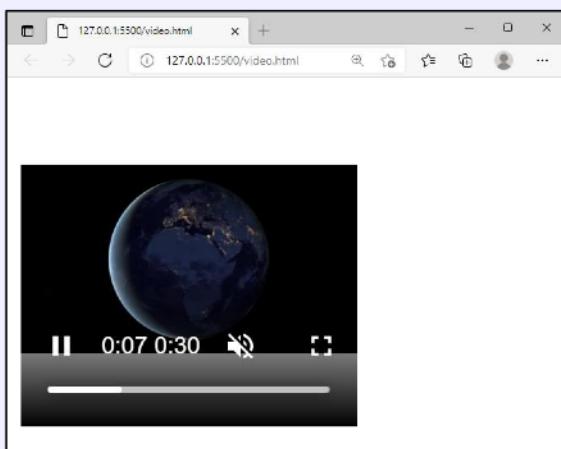
En este ejemplo vemos algunos atributos de la etiqueta `<audio>`:

- `controls`: Muestra en pantalla los controles de audio (play, pause).
- `autoplay`: Reproduce el contenido de manera automática al cargar la página.
- `loop`: Reproduce el sonido en bucle; al terminar la reproducción, comenzará de nuevo automáticamente.

Para insertar clips de vídeo se utiliza la etiqueta HTML5 `<video>`. Los archivos de vídeo podrán estar en formato mp4, ogg o WebM.

Ejemplo para insertar contenido en formato vídeo:

```
<video width="320" height="240" controls>  
  <source src="video.ogg" type="video/ogg">  
  <source src="video.mp4" type="video/mp4">  
  Lo siento, tu navegador no soporta la reproducción de  
  este vídeo.  
</video>
```



Los atributos `controls`, `autoplay` y `loop` también son aplicables a la etiqueta `<video>` con el mismo significado que en `<audio>`. Tanto en `<audio>` como en `<video>` es conveniente especificar el tamaño en pantalla del visualizador para evitar problemas de representación más tarde.

Para poder incrustar contenido en otros formatos (por ejemplo, compartido de una web externa como YouTube) se puede utilizar la etiqueta `<iframe>`. Técnicamente es una etiqueta que se emplea para insertar un contenido de otra página dentro de la actual, en formato de marco. También puede usarse para empotrar una página web dentro de la actual, aunque su uso más habitual es la importación de multimedia de sitios externos. Para esto último simplemente copiaremos el código que las propias páginas facilitan y lo personalizaremos según nuestras necesidades concretas.

Código	Resultado
<pre><html> <body> <iframe src="Image4.html" frameborder="0"> </iframe>

 <iframe src="Image5.html" frameborder="1"> </iframe> </body> </html></pre>	

Algunos de los atributos de la etiqueta <iframe> son:

- **height, width:** Dimensiones del contenido.
- **allowfullscreen (true/false) :** Indica si se puede mostrar el contenido a toda página.
- **src="URL".** Indica la fuente del contenido a visualizar.
- **name:** Nombre del marco.
- **allow:** Restricciones extra del <iframe>
 - **autoplay:** Reproducción automática.
 - **accelerometer:** Cambio de orientación en dispositivos móviles.
 - **gyroscope:** Información sobre la orientación del dispositivo en móviles.
 - **picture-in-picture:** Permite usar el modo PiP (para ver el vídeo desde otra pestaña)

Ejemplo:

```
<iframe name="video" src="https://www...." width="640"
height="480" allowfullscreen="true"
allow="autoplay;picture-in-picture"></iframe>
```

Formularios HTML

Los formularios son una de las maneras más cómodas para promover la interacción con los usuarios de una página web. Permiten enviar información desde los usuarios hacia el servidor de nuestra web, desde donde se podrá procesar. Los formularios HTML presentan una amplia variedad de elementos que permiten al usuario interactuar con el documento, incluyendo botones de diversos tipos, campos de texto, menús desplegables y más opciones. Todos los formularios quedan delimitados por la etiqueta `<form>...</form>`, y requieren de una serie de atributos para poder determinar cómo se envía información al servidor.

Atributos de un formulario:

- `name`: Nombre
- `method`: Cómo se envía la información al servidor. Dos posibles valores:
 - GET. Envía hasta 255 caracteres, y de forma pública (se ve en el navegador).
 - POST. Envía cualquier cantidad de información, de forma privada.
- `action`: Aquí se especifica qué archivo del servidor va a procesar la información enviada por el formulario. Típicamente será un archivo php.
- `target`: Dónde se muestra la posible respuesta que se reciba desde el servidor.

Elementos de un formulario:

`<input>` Crea un campo de entrada, para introducir información. Hay que especificar el tipo también, para definir las características del campo.
Ejemplos:

- `text`: Texto genérico
- `email`: Dirección de correo electrónico
- `url`: Dirección de Internet
- `number`: Número, sin limitaciones
- `date`: Fecha

- `range`: Control deslizante con números. Con este tipo de datos hay que especificar siempre los valores `min` y `max` (si no, por defecto irá de cero a cien). También el atributo `value` es interesante, porque nos posicionará la escala deslizante en el valor indicado. No presenta en pantalla el valor seleccionado.
- `datetime-local`: Fecha y hora.
- `month`: Mes.
- `time`: Hora.
- `password`: Contraseña. Escribir en este campo no genera eco en la interfaz, mostrando asteriscos en lugar del texto introducido.
- `color`: Selector de colores HTML.
- `file`: Seleccionar fichero del usuario.
- `checkbox`: Casillas de verificación.
- `radio`: Selector de una entre varias opciones.
- `reset`: Botón de "limpiado" del formulario.
- `submit`: Botón de envío del formulario.
- `button`: Botón cuya función debe definirse desde JavaScript.
- `image`: Carga una imagen que se usará como botón para enviar el formulario. Debe incluirse también el atributo `src` para que funcione.

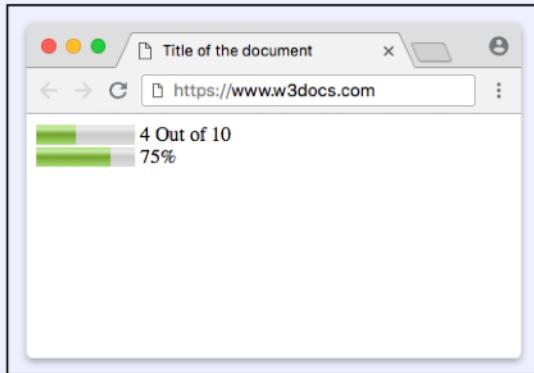
`<textarea>` Sirve para crear un campo de entrada en el que se puede insertar un texto largo, que se extienda a múltiples líneas. Con los atributos `rows` y `cols` especificaremos el tamaño en caracteres del área de texto.

`<select>` Genera listas desplegables de opciones entre las que se puede elegir. Se puede usar en conjunción con `<optgroup>` para definir grupos de opciones, siendo `<option>` el elemento que identifica cada una de esas opciones.

`<label>` Crea etiquetas para que el usuario identifique los diversos campos de un formulario.

<progress> Barra de progreso.

<meter> Indica una medida dentro de un rango; debe ser acotada, tener un inicio y un fin. Muy similar a <meter>, aunque no asociada a la finalización de una tarea; la diferencia es más bien semántica.



<range> Rango de valores predefinidos con un control deslizante para seleccionar, y con la opción de elegir secciones dentro del rango, escogiendo valores "óptimos" si así lo queremos.

HTML <Input Type="range">



<fieldset> Agrupa otros elementos de formularios. Tiene sentido en formularios extensos. Podemos añadirle el atributo <legend> para nombrar la sección concreta del formulario con una etiqueta.

A screenshot of a web form with two fieldsets. The firstfieldset is labeled "Personal Info" and contains two text input fields, one for "FirstName" and one for "LastName". The secondfieldset is labeled "Contact us" and contains two text input fields, one for "phoneNumber" and one for "Email".

Trabajar con formularios implica declararlos con la etiqueta <form> e incorporar en su interior todos los elementos que el usuario necesitará para introducir la información y enviarla al servidor. Es muy recomendable disponer de métodos para que el servidor pueda procesar la información; típicamente esto se hace con un archivo PHP. En la asignatura de Diseño de Interfaces Web nos centraremos sobre todo en el aspecto del formulario en pantalla, sin entrar en el procesamiento posterior que se realiza en el servidor.

Aunque no sea muy habitual, también hay que señalar la posibilidad de declarar como parte de un formulario elementos que no se encuentren dentro del mismo. Esto nos puede servir por diseño para tener partes del formulario fuera del espacio donde están sus demás componentes. Ejemplo:

```
<p><input type="text" name="ejemplo" form="formulario"></p>  
Esto vincula el elemento "ejemplo" con el formulario "formulario", definido en otra parte del HTML.
```

Ejemplo de formulario simple:

```
<form name="formulario" method="get" action="archivo.php">  
    <p>  
        <label for="nombre"> Nombre: </label>  
        <input type="text" name="nombre" id="nombre">  
    </p>  
    <p>  
        <label for="apellido"> Apellido: </label>  
        <input type="text" name="apellido" id="apellido">  
    </p>  
    <p>  
        <input type="submit" value="Enviar datos">  
    </p>  
</form>
```

Formularios HTML

Nombre:

Apellido:

Hemos personalizado el valor del botón “submit” con un texto.

Validación de datos en un formulario:

...

```
<input type="text" name="nombre" size="8" maxlength="10">
```

...

Formulario HTML

Texto:

Con `size` hemos definido el tamaño máximo del campo de texto en pantalla, y con `maxlength` el tamaño máximo en caracteres que puede ocupar el campo.

...

```
<input type="number" name="edad" min="18" max="99">
```

...

Valores mínimo y máximo para un campo numérico.

...

```
<input type="number" name="de5en5" min="0" max="100" step="5">
```

...

Formulario HTML

Número:

Los valores de este campo se aumentarán o reducirán de 5 en 5 al hacer clic sobre las flechas.

Otros tipos de entrada que se validan solos:

...
<label>Correo: <input type="email" name="correo"></label>

<label>Página Web: <input type="url" name="pweb"></label>

<label>Número de teléfono: <input type="tel" name="telefono"></label>

Rangos (escala deslizante):

...
<input type="range" name="porcentaje" min="1" max="100" value="75">

Formulario HTML

Rango: 

Escala de 0 a 100, valor por defecto en 75.

Botones de radio:

```
<form name="formulario" method="get" action="procesar.php">  
<p>  
    ¿Te gusta este formulario?  
</p>  
<p>  
    <label><input type="radio" name="resp" value="1" checked> Sí </label>  
</p>  
<p>  
    <label><input type="radio" name="resp" value="2"> No </label>  
</p>  
<p>  
    <label><input type="radio" name="resp" value="3"> NS/NC </label>  
</p>  
<p>  
    <input type="submit" value="Enviar">  
</p>  
</form>
```

Formulario HTML

¿Te gusta este formulario?

Sí

No

NS/NC

Valor por defecto: 1- Sí (atributo `checked`); al servidor se le enviará 1, 2 o 3 según la respuesta escogida.

Casillas de verificación:

...

```
<p> Aficiones: </p>
<p>
    <label> <input type="checkbox" name="op1" value="1" checked>
    Informática</label>
</p>
<p>
    <label> <input type="checkbox" name="op2" value="2">
    Deporte</label>
</p>
<p>
    <label> <input type="checkbox" name="op3" value="3">
    Cine</label>
</p>
```

...

Formulario HTML

Aficiones:

- Informática
- Deporte
- Cine

Fechas (presenta calendario en pantalla):

...

```
<input type="date" name="fecha" value="2000-12-31">
```

...

Fecha con valor por defecto seleccionado, 31/12/2000

Formulario HTML

31 / 12 / 2000

Colores (presenta selector hexadecimal):

...

```
<input type="color" name="colores" value="#F04B37">
```

...

Color con valor por defecto (opcional) seleccionado.

Formulario HTML

Color:

Botones de envío con imágenes:

...

```
<p><input type="image" src="boton.jpg" width="300" height="500">
```

...

Párrafos de texto:

...

```
<p>
```

```
  <label> Texto largo: <textarea name="texto" cols="40" rows="5"></textarea></label>
```

```
</p>
```

...

Formulario HTML

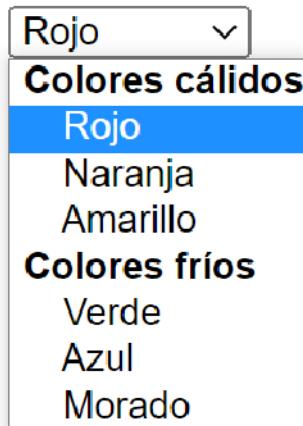
Texto largo:

Lista desplegable básica:

```
<form name="formulario" method="get" action="archivo.php">  
<p>  
    <label for="desplegable"> Color favorito:</label>  
    <select name="desplegable" id="desplegable">  
        <optgroup label="Colores cálidos">  
            <option value="red">Rojo</option>  
            <option value="ora">Naranja</option>  
            <option value="yel">Amarillo</option>  
        </optgroup>  
        <optgroup label="Colores fríos">  
            <option value="gre">Verde</option>  
            <option value="blu">Azul</option>  
            <option value="pur">Morado</option>  
        </optgroup>  
    </select>  
</p>  
<p>  
    <input type="submit" value="Enviar">  
</p>  
</form>
```

Formulario HTML

Color favorito:



Si queremos hacer que aparezca un texto no seleccionable indicando que se debe elegir una opción, y no queremos poner una de las opciones del <select> marcada por defecto:

```
<form name="comida" method="get" action="/action_page.php">
    <label for="alim">¿Qué quieres para cenar? </label>
    <select name="alim" id="alim" required>
        <option value="" checked hidden> --Elige una
        opción</option>

        <option value="pat">Patatas</option>
        <option value="tom">Tomates</option>
        <option value="ceb">Cebollas</option>
        <option value="pim">Pimientos</option>
    </select>
    <br><br>
    <input type="submit" value="Enviar">
</form>
```

Formulario HTML

¿Qué quieres para cenar?

Lista con sugerencias (autocompletear conforme se escribe):

```
<form name="formulario" method="get" action="archivo.php">  
    <label for="colores">Elige tu color favorito:</label>  
    <input list="colores" name="color" id="color">  
    <datalist id="colores">  
        <option value="Rojo">  
        <option value="Naranja">  
        <option value="Amarillo">  
        <option value="Verde">  
        <option value="Azul">  
        <option value="Violeta">  
    </datalist>  
    <p>  
        <input type="submit" value="Enviar">  
    </p>  
</form>
```

Formulario HTML

Elige tu color favorito:

Naranja

Envío del formulario:

<form method="get"> Los datos del formulario, en formato de parejas atributo/valor se envían en la propia URL, siendo visibles para el usuario y con un tamaño limitado, de hasta 255 caracteres.

<form method="post"> Los datos del formulario se envían en el cuerpo de la solicitud HTML, no visibles para el usuario y sin un límite de tamaño.

Ejemplo de presentación de resultados de un formulario en un archivo PHP:

```
...
<form name="f1" method="get" action="archivo.php">
    <p>
        <label>Ejemplo:<input type="text" name="ejemplo"> </label>
    </p>
    <p>
        <input type="submit" value="Enviar">
    </p>
</form>
...
→archivo.php
...
<p>
<?php
print('Valor introducido: ' . $_GET['ejemplo']);
?>
</p>
...

```

Si en este ejemplo hubiéramos escrito `method="post"` en el formulario, en el archivo PHP correspondiente deberíamos haber usado `$_POST['ejemplo']`.

Atributos globales aplicables en formularios HTML

Dentro de los elementos que conforman un formulario existen numerosos atributos globales que se pueden utilizar para producir diferentes efectos sobre los campos correspondientes. Algunos ejemplos de esos atributos son:

- `disabled`: Deshabilita el elemento.
- `readonly`: El contenido del elemento no se puede modificar.

- `placeholder="texto"`: Presenta un texto de "ayuda" que indica al usuario qué debe escribir en el campo.
- `required`: Campo obligatorio. Si ese campo no está relleno el formulario no puede enviarse.
- `autofocus`: El elemento que tenga este atributo booleano tendrá el cursor (foco) en el momento en que se cargue el documento.
- `spellcheck="true | false"`: Comprueba la ortografía y gramática del texto introducido. Por defecto está activo para los campos de texto.
- `pattern`: Define expresiones regulares que se aplicarán al texto introducido para comprobar si se encuentran en el rango de valores válidos para el campo. Nos permite personalizar la validación del elemento de formulario. Puede usarse en conjunción con el atributo `title` para indicar al usuario qué tipo de contenido debería introducirse.

Expresiones regulares en HTML

Muy utilizadas en Informática en general, las expresiones regulares son muy interesantes para validar elementos dentro de un formulario. Aunque esta tarea también se puede (y se debe) hacer en el backend, si no permitimos la entrada de datos erróneos en nuestra aplicación será más fácil manejarlos. Una expresión regular es un patrón básico que se intentará localizar en cadenas de texto, en nuestro caso en los elementos de entrada de los formularios. La sintaxis de las expresiones regulares en HTML es muy similar a la de JavaScript, pudiendo crear patrones muy complejos que se ajusten perfectamente a las características de la entrada que necesitemos.

Ejemplos de expresiones regulares:

[0-9] Cualquier número entre 0 y 9

[A-Z] Cualquier letra mayúscula

[A-Za-z]{5} Cualquier cadena de exactamente 5 caracteres constituida exclusivamente por letras mayúsculas y/o minúsculas.

[0-9]{8}[A-Z]{1} 8 dígitos seguidos de exactamente 1 letra mayúscula

[0-9]{1,3}[A-Za-z]{1,2} Entre 1 y 3 dígitos, seguidos de 1 o 2 letras mayúsculas y/o minúsculas.

[a-zA-Z]{3,} Cadena de al menos 3 caracteres, sin límite de longitud, con letras mayúsculas y/o minúsculas.

[^A-Z] Un carácter que no sea una letra mayúscula.

[0-9]+ Cadena de al menos un dígito. No hay longitud máxima.

[a-z][A-Z]* Una letra minúscula seguida de 0 o más caracteres en mayúscula.

[0-9]{5}[a-z]? Una cadena de exactamente 5 dígitos, seguida de 0 o 1 letra minúscula.

[0-9]{3}[a-z]{2}? Una cadena de exactamente 3 dígitos, seguida por nada o una cadena de exactamente dos letras minúsculas.

([0-9] | [A-Z])[a-z]{3} Un dígito o una letra mayúscula (pero no ambas cosas), seguido por exactamente 3 letras minúsculas.

[0-9]{8}(-)?([13579] | [A-Z]) una cadena de exactamente 8 dígitos, con la presencia opcional del carácter guión, y finalizada por una letra mayúscula o un número impar.

[a-z]{2,}(\.|\?) Una cadena de letras minúsculas de al menos dos caracteres de longitud, seguida por el carácter del punto o el carácter del signo de interrogación. En este ejemplo el carácter "\ sirve para que se interprete de forma literal el símbolo al que precede, y no como parte de una expresión regular.

. [0-9]. Cualquier carácter seguido de un dígito y después otra ocurrencia de cualquier carácter.

Abreviaturas en expresiones regulares:

- \d Un dígito
- \D Cualquier carácter que no es un dígito. Equivale a [^0-9]
- \w Una letra (mayúscula o minúscula), o un dígito entre 0 y 9, o el carácter de subrayado (_). Equivale a [a-zA-Z_0-9]
- \W Cualquier carácter no alfanumérico. Es la opuesta a la expresión anterior
- \s Espacio en blanco o tabulación
- \S Cualquier carácter que no sea ni el espacio en blanco ni la tabulación

- `\t` Carácter de tabulación. En formularios solamente puede insertarse copiando y pegando, porque pulsar la tecla de tabulación en un campo de formulario simplemente moverá el cursor al siguiente campo.
- Para caracteres especiales podemos usar su número en Unicode. Por ejemplo, `\x24` equivale al símbolo del dólar (\$)
- `\.` Carácter del punto

(?=) *Lookahead assertion*. Indicamos que en la expresión completa del texto introducido por el usuario debe aparecer, en algún punto, la expresión dentro del paréntesis. Si, por ejemplo, vamos a incluir requerimientos de longitud de cadena, es mejor ponerlos al principio de la expresión regular y rematar el *lookahead assertion* con el símbolo \$. Esto evita ambigüedades porque representa el final de la subexpresión regular y no se mezcla con otras posibles *assertions* que hayamos incluido en la expresión regular.

Ejemplo: Validación de un campo que debe tener al menos 6 caracteres de longitud e incluir mayúsculas, minúsculas y dígitos en su composición.

(?= {6,} \$) (=? .* [A-Z]) (=? .* [a-z]) (=? .* [0-9]) .+

Alternativa: (=? .* [A-Z]) (=? .* [a-z]) (=? .* [0-9]) (. {6,})

(?!) *Negative lookahead*. Es lo contrario de la expresión anterior, ya que detecta patrones que no deben cumplirse en la expresión introducida por el usuario para que se considere válida.

Ejemplo: Validación de un campo con 5 dígitos, pero en el que no pueden existir dos ceros ni dos unos de forma consecutiva.

(?! .* (00|11) .*) ([0-9]{5})

Al igual que con JavaScript podemos poner el carácter \$ como indicador del fin de una cadena. Por ejemplo, esta expresión representa una cadena de 8 caracteres compuesta por letras minúsculas y números (debe aparecer al menos un carácter de cada tipo), pero que no puede terminar en 0 ni en 5.

(=? .* [a-z]) (=? .* [0-9]) (?! .* (0|5) \$) [a-z0-9]{8}

Cuando tenemos campos de un formulario con algún tipo de validación, podemos personalizar el mensaje que se muestra en caso de introducir valores incorrectos. Aquí tenemos un ejemplo que hace uso de la propiedad `oninvalid`, que indica qué hacer si se introduce un valor no válido en el campo, y el método `setCustomValidity()` de JavaScript:

```
<form name="codigo" method="get" action="/action_page.php">

    <label for="codpos">Introduce tu código postal:

        <input type="text" name="codpos" id="codpos" required
            pattern="\d{5}" placeholder="Código Postal"
            oninvalid="this.setCustomValidity('Por favor,
            introduce un número de 5 dígitos')"
            oninput="this.setCustomValidity('')" maxlength=5
            size=10/>

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

</form>
```

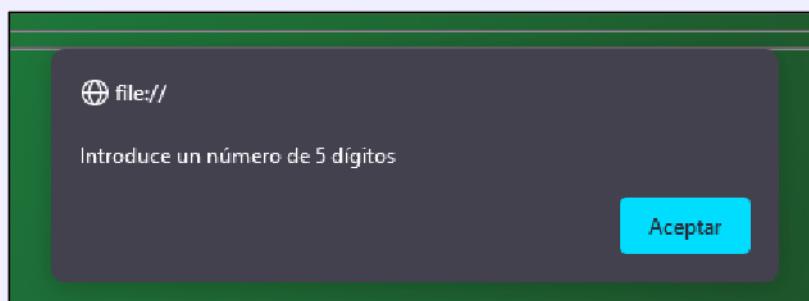
Formulario HTML

Introduce tu código postal:

! Por favor, introduce un número de 5 dígitos

Otra forma más simple (aunque menos elegante) de hacer lo mismo es utilizar un mensaje con el método `alert` de JavaScript, como se muestra en el siguiente fragmento de código para el mismo `<input>`. Esto mostrará una alerta que habrá que aceptar antes de poder volver a introducir información en el campo.

```
<form name="codigo" method="get" action="/action_page.php">  
  
    <label for="codpos">Introduce tu código postal:  
  
        <input type="text" name="codpos" id="codpos" required  
            pattern="\d{5}" placeholder="Código Postal"  
            oninvalid="alert('Introduce un número de 5 dígitos');"  
            maxlength=5 size=10/>  
  
        <input type="submit" value="Enviar">  
  
</form>
```



Validación de formularios

Aunque no es el objetivo de este módulo, es importante destacar que cualquier formulario con el que los usuarios puedan comunicarse con el servidor debe estar perfectamente validado para permitir su envío. No solamente debemos establecer validaciones sobre los campos concretos (ya sea con el tipo de `<input>` o mediante expresiones regulares en el atributo `pattern`), sino que desde el propio código PHP deberemos asegurar que no hay problemas con el código introducido por los usuarios.

Así, por ejemplo, si no controlamos ese código que se envía, sería técnicamente posible que un hipotético atacante introdujera código en formato de script dentro de un campo de un formulario y eso permitiese la ejecución de código no deseado en nuestro servidor. Una forma de evitar que eso ocurra es usar un código parecido a lo siguiente:

...

```
<?php

$nombre = $apellido = $direccion = $telefono = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $nombre = validar_input($_POST["nombre"]);
    $apellido = validar_input($_POST["apellido"]);
    $direccion = validar_input($_POST["direccion"]);
    $telefono = validar_input($_POST["telefono"]);
}

function validar_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

?>

...
```

En este ejemplo se hace uso de las siguientes funciones:

- **trim**: Elimina los espacios al principio y al final de una cadena.
- **stripslashes**: Elimina las contrabarras (\) de una cadena. Si hay dos contrabarras consecutivas solamente elimina una de ellas.
- **htmlspecialchars**: Sustituye caracteres no alfanuméricos por su equivalente en código HTML. Por ejemplo, sustituirá el símbolo "<" por "<". Esto evitirá que un usuario malintencionado pueda insertar código en un campo de formulario.

Por supuesto, será necesario incluir el código necesario para manejar los datos introducidos por los usuarios y en su caso presentar una información sobre el

resultado de las operaciones realizadas. Una manera (extremadamente básica e incompleta) de hacer esto último sería la siguiente:

```
... (código HTML para mostrar contenido)
<?php

...
echo $nombre;
echo "<br>";
echo $apellido;
echo "<br>";
echo $direccion;
echo "<br>";
echo $telefono;
echo "<br>";

...
?>
```