

HTML 5

(parte 1)

enero 2019

Prof. Ainhoa Iglesias



Este obra está bajo una licencia de [Creative Commons Reconocimiento - NoComercial - CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-nc-sa/4.0/)

ÍNDICE:

HTML 5	3
PARTE 1: INTRODUCCIÓN	4
1. HACIENDO HISTORIA.....	5
2. ETAPAS DE UN ESTÁNDAR W3C	6
3. OBJETIVOS DE HTML 5.....	7
3. DECLARACIÓN Y CODIFICACIÓN	8
4. NORMAS Y RECOMENDACIONES.....	9
5. VALIDACIÓN	11
6. LO QUE DESAPARECE	12
PARTE 2: ELEMENTOS PRINCIPALES.....	13
1. GENERALES	14
2. ENLACES:	20
3. IMÁGENES:	22
4. TÍTULOS (HEADINGS):	24
5. LISTAS:.....	26
6. TABLAS:	29
7. FORMULARIOS	31
8. ELEMENTOS META	39
9. ELEMENTOS LINK	41

HTML 5

PARTE 1: INTRODUCCIÓN

1. HACIENDO HISTORIA



W3C:

1997: publica HTML 4 como recomendación.

1997-2006: centra su esfuerzo en el desarrollo de XHTML 1.

WHATWG (Web Hypertext Application Technology Working Group):

Junio'04 – Marzo'07: trabajando en una evolución del HTML (no XHTML) al margen del W3C.



W3C:

Marzo'07:

- crea un Grupo de trabajo (Working Group) para desarrollar HTML 5.
- crea otro Grupo de trabajo para desarrollar XHTML 2.

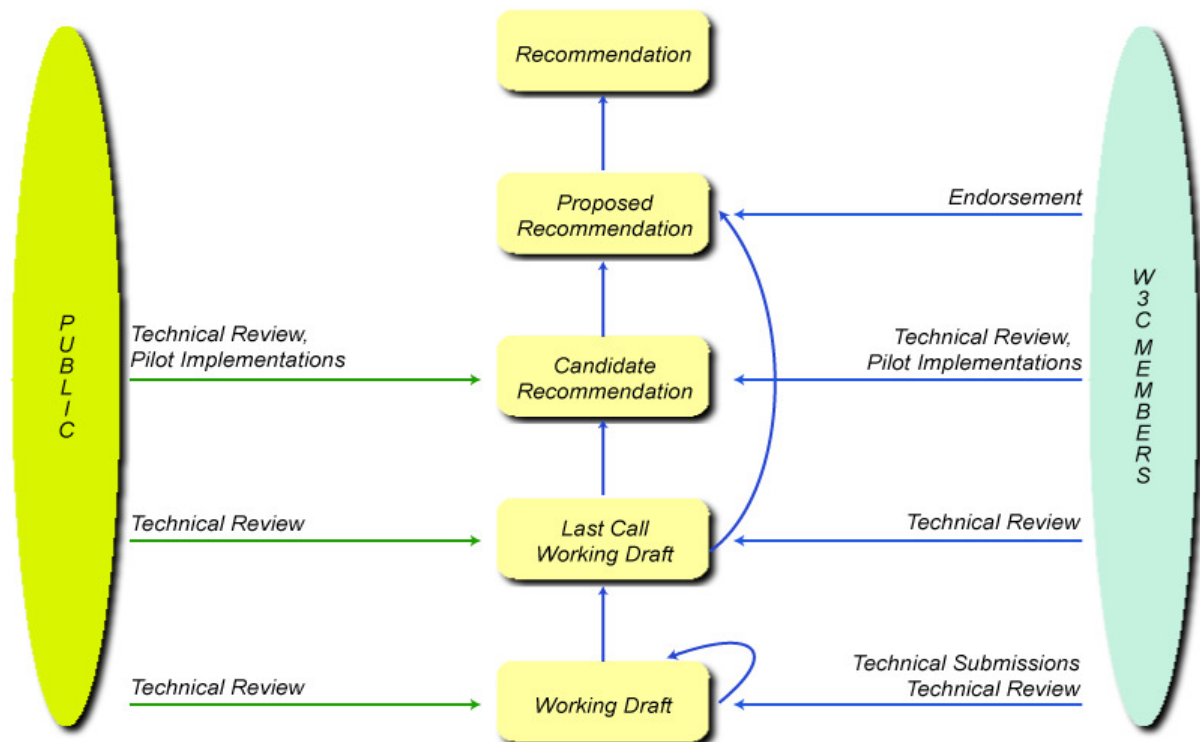
Ambos grupos trabajan en paralelo.

Febrero'09:

- Se anuncia que el Grupo de trabajo de XHTML 2 cesará su actividad a finales de 2009.
- Se aumentarán los recursos para el Grupo de trabajo de HTML 5.
- El trabajo de mantenimiento sobre XHTML 1.1 continuará, para arreglar bugs.

En 2009 nos preguntábamos...¿cuándo se aprobará HTML 5 como estándar?

2. ETAPAS DE UN ESTÁNDAR W3C



HITOS HTML 5

Abril'07: especificaciones HTML 5 y Web Forms 2.0 adoptadas como bases para revisión.

Noviembre'07: (principios de diseño HTML) 1º borrador de trabajo público.

Febrero'08: 1º borrador de trabajo público.

Marzo'09*: Última llamada borrador de trabajo. (los editores estiman Octubre'09)

Junio'09: Candidato a Recomendación.

Junio'10: Propuesto a Recomendación.

Septiembre'10: HTML5 es Recomendación.

FECHA ESTIMADA

Todas las estimaciones apuntaban al año 2012...

Pero finalmente fue en octubre de 2014¹.

Y se sigue desarrollando²

¹ <http://www.w3.org/blog/news/archives/4167>

² <http://w3c.github.io/html/>

3. OBJETIVOS DE HTML 5

Los objetivos de HTML 5 son:

- Proveer buen soporte para contenidos existentes y aplicaciones web.
- Optimizar la retro-compatibilidad (con HTML 4, XHTML 1)
- Nuevas APIs.
- Asegurar la interoperabilidad.
- Definir de manera precisa el comportamiento de los Agentes de Usuario.
- Especificación formal del manejo de errores.
- Evolución (no revolución).

MANEJO DE ERRORES

- HTML no es XML.
- HTML no es SGML.
- La mayoría del contenido HTML no está bien formado (de acuerdo a XML).

Y a pesar de los fallos, las páginas web se visualizan y evolucionan!!

3. DECLARACIÓN Y CODIFICACIÓN

El doctype se reduce a la mínima expresión:

```
<!DOCTYPE html>
```

Lo mismo con la codificación de caracteres:

```
<meta charset="utf-8">
```

Por lo tanto, si en nuestra plantilla base antes usábamos:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-type" content="text/html;
charset=utf-8" />
    <title>XHTML 1.0</title>
  </head>
  <body>
    <h1>Hola</h1>
  </body>
</html>
```

Ahora lo vamos a reducir a:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>HTML5</title>
  </head>
  <body>
    <h1>Hola</h1>
  </body>
</html>
```


4. NORMAS Y RECOMENDACIONES

Los documentos deben ser "gramaticalmente correctos":

Todos los elementos deben estar correctamente anidados.

Correcto:

Incorrecto:

```
<p>Hola, <strong>amigo</strong></p> <p>Hola, <strong>amigo</p></strong>
```

Los nombres de elementos y atributos, mejor en minúsculas:

Se recomienda que el código html se escriba en minúsculas.

 mejor que o que

Los valores de los atributos, mejor entre comillas:

Todos los valores de atributos es mejor que vayan entrecomillados (dobles o simples):

Correcto:

```

```

Incorrecto:

```

```

Todas las etiquetas, mejor cerrarlas:

La mayor parte de los elementos HTML tienen etiqueta de cierre, pero existen algunos que no lo tienen.

En ese caso, se recomienda que se autocierren, con una "/" al final:

```
<p>un párrafo</p>
<li>elemento de lista</li>
<input />
<img />
<meta />
<hr />
<br />
```

Atributos, mejor con valor:

Se recomienda que todos los atributos lleven siempre un valor (en vez de "minimizar" los atributos).

Atributo con valor:

```
<dl compact="compact">
<input ... readonly="readonly" />
<input ... disabled="disabled" />
<option value="" selected="selected" />
<input ... checked="checked" />
```

Atributo minimizado:

```
<dl compact>
<input ... readonly />
<input ... disabled />
<option ... selected />
<input ... checked />
```

Ampersands (&'s) en las URL³:

Para evitar errores en la generación de una dirección, se recomienda codificar el símbolo del ampersand (&) mediante su equivalente "&".

Correcto:

```
<a href="index.php?op=1&amp;lang=en">
```

Incorrecto:

```
<a href="index.php?op=1&lang=en">
```

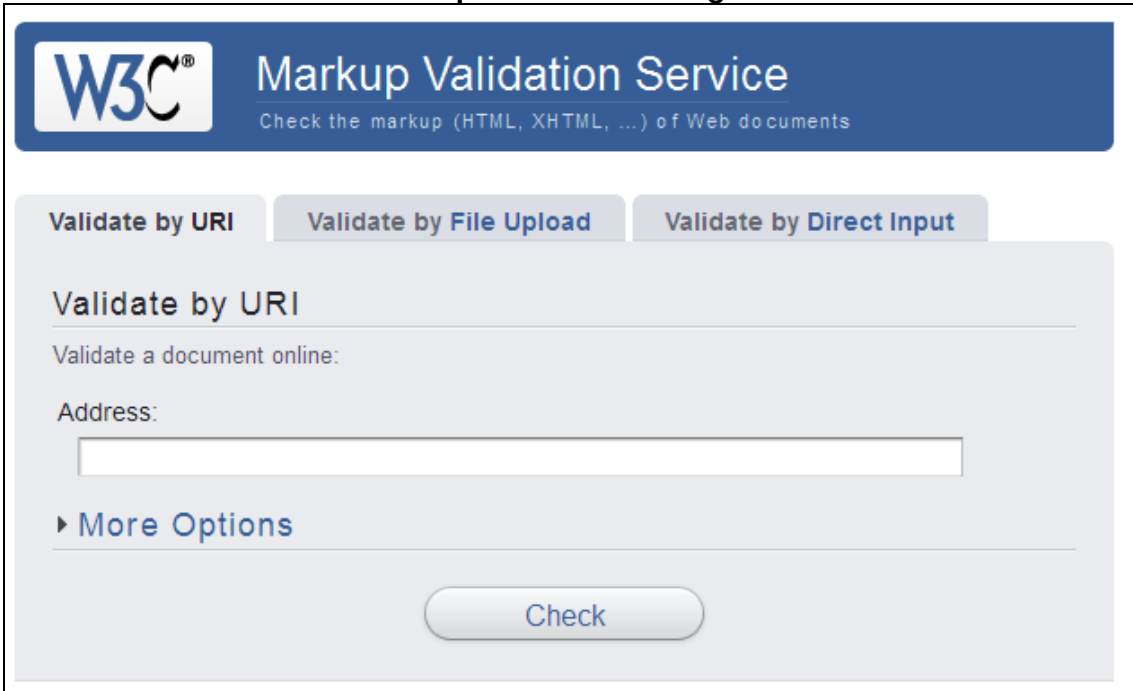
³ Puedes consultar más información sobre el uso de ampersand en las siguientes direcciones:
<http://www.htmlhelp.com/tools/validator/problems.html#amp>
<http://www.w3.org/QA/2005/04/php-session>

5. VALIDACIÓN

Cuando escribimos un documento de texto, normalmente pasamos un corrector ortográfico para que nuestro texto esté libre de faltas de ortografía. Con el código de nuestras páginas web debemos hacer lo mismo.

Para ello, utilizaremos el validador de código que el W3C pone a nuestra disposición:

<http://validator.w3.org/>



The image shows the W3C Markup Validation Service interface. At the top, there is a blue header with the W3C logo and the text "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents". Below the header, there are three tabs: "Validate by URI", "Validate by File Upload", and "Validate by Direct Input". The "Validate by URI" tab is selected. Under this tab, there is a section titled "Validate by URI" with the text "Validate a document online:". Below this, there is a label "Address:" followed by a text input field. Below the input field, there is a link "More Options". At the bottom of the form, there is a "Check" button.

6. LO QUE DESAPARECE

Los siguientes elementos y atributos no existen dentro de la especificación HTML 5 (algunos ya habían desaparecido en XHTML)

Elementos:

acronym
applet
basefont
big
center
dir
font
frame
frameset
isindex
noframes
noscript
s
strike
tt
u

Atributos:

abbr
archive
axis
charset
classid
codetype
declare
header
name
nohref
profile
rev
scheme
standby
summary
target
valuetype
version

PARTE 2: ELEMENTOS PRINCIPALES

1. GENERALES

Comentarios:

Si necesitamos escribir un comentario en el código fuente de la página, pero que no aparezca por pantalla, debemos utilizar las siguientes etiquetas:

Ejemplo:

```
<!-- Esto es un comentario -->
```

Párrafos:

Mediante los párrafos, estructuraremos la mayor parte de los contenidos web:

Ejemplo:

```
<p>este es un párrafo</p>
<p>este es otro párrafo</p>
```

este es un párrafo

este es otro párrafo

Salto de línea:

Se puede dar el caso en el que necesitemos saltar de línea dentro de un párrafo:

Ejemplo:

```
<p>
Imagine there's no heaven<br />
It's easy if you try
</p>
```

Imagine there's no heaven

It's easy if you try

Mediante las propiedades CSS conseguiremos eliminar el uso abusivo de saltos de línea para separar elementos.

No podemos utilizar dos elementos `
` seguidos sólo para simular un párrafo.

Énfasis:

Cuando queremos destacar un texto, solemos mostrarlo en negrita y, al leerlo, lo hacemos con otro tono de voz.

Antiguamente utilizábamos los elementos `` e `<i>` (bold e italic) para tal efecto, pero eran simplemente marcadores de diseño. Cuando un navegador parlante leía ese texto, no cambiaba la entonación, porque para él, no tenía ningún significado especial.

Por eso, debemos utilizar los elementos semánticos `` (importante) y `` (muy importante).

El estilo por defecto de estos elementos (cómo se muestra en los navegadores) es la negrita para `` y la itálica (cursiva) para ``.

Ejemplo:

```
<p>
<strong>Aviso:</strong> No olvide utilizar un <em>DNI válido</em>.
</p>
```

Aviso: No olvide utilizar un *DNI válido*.

Bloque de cita:

Cuando queremos mostrar un bloque de texto citado textualmente, debemos utilizar este elemento:

Ejemplo:

```
<blockquote cite=" http://www.quijote.org/QR101.htm">
<p>En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha
mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga
antigua, rocín flaco y galgo corredor.</p>
</blockquote>
```

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

No se debe utilizar este elemento simplemente para utilizar la sangría que proporcionan por defecto, si no estamos citando un texto.

Citas:

Para citas en línea y cortas, utilizaremos el elemento `<q>` para el texto y `<cite>` para el autor. Los navegadores suelen mostrar `<cite>` en cursiva.

Ejemplo:

```
<p>Como dijo <cite>Hamlet</cite>: <q>ser o no ser, esa es la
cuestión</q>.</p>
```

Como dijo *Hamlet*: "ser o no ser, esa es la cuestión".

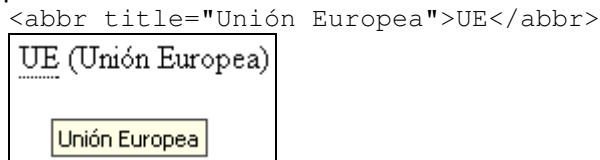
Abreviaturas

Para marcar cualquier tipo de abreviatura, sigla o acrónimo, se utiliza el elemento `<abbr>`.

Si queremos expandir su significado, utilizaremos el atributo `title*`, con el significado correspondiente.

* El atributo `title` se puede utilizar en la práctica totalidad de los elementos HTML (enlaces, campos de formulario...)

Ejemplo:



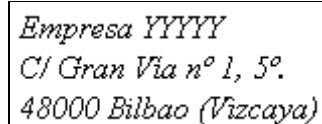
Dirección:

Para marcar la información de contacto con el autor/responsable de la página, debemos utilizar el elemento `<address>`.

Normalmente, los navegadores muestran el texto incluido en este elemento en cursiva, pero mediante CSS podremos variar su aspecto:

Ejemplo:

```
<address>
<p>Empresa xx<br />
C/ Gran Vía nº 1, 5º.<br />
48000 Bilbao (Vizcaya)</p>
</address>
```



Insertado y borrado:

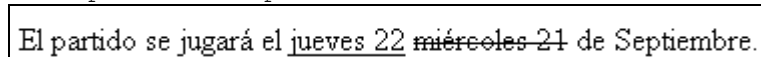
Si nos interesa que los visitantes conozcan las actualizaciones y cambios de un documento, podemos hacer uso de los elementos `<ins>` y ``.

`` sirve para marcar el texto que se ha borrado del documento e `<ins>`, para marcar el texto nuevo que se ha insertado.

Además podemos hacer uso del atributo `cite` para indicar la URL de un documento donde se explica el por qué de ese insertado o borrado, y del atributo `datetime` para definir la fecha de la modificación (en formato AAAAMMDD).

Ejemplo:

```
<p>El partido se jugará el <ins>jueves 22</ins><del>miércoles 21</del>
de Septiembre.</p>
```



Preformateado, código, definición, ejemplo, variable y entrada:

```
<pre>
```

Mediante el elemento `<pre>` podemos incluir texto respetando los espacios en blanco y los saltos de línea, sin necesidad de crearlos. Es muy útil cuando tenemos que escribir contenido con indentaciones:

Ejemplo:<pre>

```
&lt;div id="importante"&gt;
  &lt;p&gt;
    Aviso: Todos los campos son obligatorios.
  &lt;/p&gt;
&lt;/div>
```


<code>

El elemento <code> sirve para indicar que el texto que estamos mostrando pertenece a un código informático:

Ejemplo:

```
<code>
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
</code>
```

```
class HelloWorld { public static void main(String[] args) {
    System.out.println("Hello World!"); } }
```

<kbd>

El elemento <kbd> lo utilizaremos para indicar que el texto mostrado pertenece a una tecla o combinación de teclas que tendremos que introducir por nuestro teclado.

Ejemplo:

<p>Si pulsas la tecla <kbd>F1</kbd> podrás acceder a la ayuda del programa</p>

Si pulsas la tecla F1 podrás acceder a la ayuda del programa

Mediante CSS podrás conseguir un efecto gráfico tan sorprendente como éste:

Si pulsas la tecla **F1** podrás acceder a la ayuda del programa

<dfn> <var> <samp>

También disponemos del elemento <dfn> para indicar un término de definición, el elemento <var> para definir una variable y el elemento <samp> para definir un código informático de ejemplo.

Orden del tabulador:

Si queremos modificar el orden de navegación mediante el tabulador, podremos hacer uso del atributo tabindex.

Ejemplo:

Imaginemos que tenemos un formulario (mal) maquettato con tablas:

```
<table border="1">
<tr>
<td>Nombre:<br /><input type="text" name="nom" /></td>
<td>Población:<br /><input type="text" name="pob" /></td>
</tr>
```

```

<tr>
<td>Apellidos:<br /><input type="text" name="apel" /></td>
<td>CP:<br /><input type="text" name="cp" /></td>
</tr>
<tr>
<td>Dirección:<br /><input type="text" name="dir" /></td>
<td>Teléfono:<br /><input type="text" name="tfn" /></td>
</tr>
</table>

```

Como el orden de navegación sigue siempre la secuencia del código fuente, la secuencia será: nombre, población, apellidos, cp, dirección y teléfono

Nombre: <input type="text"/>	1	Población: <input type="text"/>	2
Apellidos: <input type="text"/>	3	CP: <input type="text"/>	4
Dirección: <input type="text"/>	5	Teléfono: <input type="text"/>	6

La secuencia más lógica hubiera sido otra: nombre, apellidos, dirección, población, cp y teléfono.

Para modificarla, bastaría con añadir el atributo `tabindex` en cada elemento del formulario:

```

<table border="1">
<tr>
<td>Nombre:<br /><input type="text" name="nom" tabindex="1" /></td>
<td>Población:<br /><input type="text" name="pob" tabindex="4" /></td>
</tr>
<tr>
<td>Apellidos:<br /><input type="text" name="apel" tabindex="2" /></td>
<td>CP:<br /><input type="text" name="cp" tabindex="5" /></td>
</tr>
<tr>
<td>Dirección:<br /><input type="text" name="dir" tabindex="3" /></td>
<td>Teléfono:<br /><input type="text" name="tfn" tabindex="6" /></td>
</tr>
</table>

```

Nombre: <input type="text"/>	1	Población: <input type="text"/>	4
Apellidos: <input type="text"/>	2	CP: <input type="text"/>	5
Dirección: <input type="text"/>	3	Teléfono: <input type="text"/>	6

Atajos de teclado:

Podemos implementar atajos de teclado para nuestros enlaces, así como existen en todos los programas que utilizamos. Basta con utilizar el atributo `accesskey` y darle un valor (tecla) de letra o número que queramos utilizar como atajo.

Ejemplo:

```
<a href="index.html" accesskey="1">Página de inicio</a>
```

La combinación de teclas varía de unos navegadores a otros: ALT + Tecla de atajo de teclado o ALT + Tecla de atajo de teclado + ENTER...

Es recomendable utilizar sólo los números para evitar posibles interferencias con las letras que sirven como atajo de teclado en los navegadores y ayudas técnicas, salvo la letra "S" para implementar el salto al contenido. También se recomienda utilizar el número "1" para la página de inicio.

2. ENLACES:

Los enlaces se implementan mediante el elemento `a` y se compone, básicamente, de ruta y texto del enlace:

Si la página que vamos a enlazar se encuentra en el mismo directorio que la página en la que estamos, bastará con escribir el nombre del fichero. Si está alojada en un subdirectorio, entonces utilizamos una barra (/) para indicar el camino y si se encuentra en un directorio de un nivel superior, utilizamos dos puntos y una barra (../)

Ejemplo:

```
<a href="ruta">texto del enlace</a>
<a href="mipagina.html">página personal</a>
<a href="personal/mipagina.html">mi página</a> (en subdirectorio)
<a href="../mipagina.html">mi página</a> (en nivel superior)
```

Podemos subir o bajar de nivel las veces que sea necesario:

Ejemplo:

```
<a href="personal/io/mio/mipagina.html">mi página</a>
<a href="../../mipagina.html">mi página</a>
```

Enlace a una página externa:

Durante años, la costumbre más extendida para implementar los enlaces a páginas externas era utilizar el atributo `target="_blank"`.

Sin embargo, en la actualidad, esta práctica está desaconsejada, puesto que los navegadores utilizan mecanismos para no abrir en una nueva ventana (sino en una nueva pestaña) o incluso bloquean esta orden.

Se suelen utilizar soluciones javascript para ello (marcando el enlace como externo con el atributo `rel="external"` y aplicando una función javascript sobre este atributo).

Podemos utilizar el atributo `title` para ofrecer una pequeña descripción de la página a la que vamos a enlazar. No tendrá mucho sentido si el texto del título es idéntico al texto del enlace.

Correcto:

```
<a href="http://www.yahoo.com">Yahoo</a>
<a href="http://www.yahoo.com" title="Yahoo: noticias, e-mail, fotos,
grupos, etc..."> Yahoo</a>
```

Incorrecto:

```
<a href="http://www.yahoo.com" target="_blank"> Yahoo</a>
<a href="http://www.yahoo.com" title="Yahoo"> Yahoo</a>
```

Es muy importante que nuestro enlace comience por `http://` (o `https://`). De lo contrario, no funcionará.

Enlaces dentro de una página:

Para enlaces dentro de la misma página o dentro de otra página, utilizamos las anclas.

En el punto donde queremos saltar, insertamos un “ancla”, mediante el atributo “id” y después implementamos un enlace que lleve a ese id.

En el enlace, tenemos que poner una almohadilla delante del nombre del id.

Ejemplo:

```
<a href="#ancla1">Canción 2</a>
<a href="noticias.html#ancla1">Canción 2</a>
.....
<a id="ancla1"></a>
```

Idioma de destino de un enlace:

Si un enlace apunta a una página en un idioma diferente al que estamos, podemos indicar ese lenguaje mediante el atributo hreflang:

Ejemplo:

```
<a href="http://www.nasa.gov" hreflang="en">NASA</a>
<a href="http://www.paris.fr/" hreflang="fr">Ayuntamiento de París</a>
```

3. IMÁGENES:

Las imágenes se implementan mediante el elemento `` y se compone, básicamente, de ruta y texto alternativo:

Ejemplo:

```

```

Los atributos `width` y `height` (altura y anchura) no son atributos de diseño, sino que son propiedades intrínsecas de la imagen. No son obligatorios y hace tiempo se recomendaba ponerlos para que el navegador ahorrara tiempo a la hora de dibujar la página, puesto que reservaba físicamente el espacio determinado por esos valores. Sin embargo, a día de hoy, salvo excepciones, esto impediría que las imágenes se adaptasen al tamaño de pantalla.

Correcto:

```

```

Incorrecto:

```


```

En algunas comunidades de desarrolladores, está muy extendida la práctica de añadir el atributo `title` a las imágenes, con el mismo valor que el del atributo `alt`. ¿por qué?

Esto viene de hace muchos años atrás. En su día ocurría lo siguiente:

Si sólo utilizamos el atributo `alt` y acceder a la página mediante el navegador Internet Explorer, al pasar el ratón por encima de una imagen aparecerá una barra (tooltip) con el texto alternativo de esa imagen. Sin embargo, si visitamos la página con Chrome, Mozilla Firefox u Opera, por ejemplo, no aparecerá ese tooltip. Es entonces cuando se nos ocurre utilizar el atributo `title` y, de esta manera, también aparecerá en estos navegadores. Sin embargo, estamos yendo en contra de la especificación.

El comportamiento del atributo `alt` indica que no debe aparecer en forma de tooltip, sirve para ofrecer un contenido alternativo e informar al usuario del significado de la imagen, en el caso de que no se cargue, la ruta sea incorrecta, utilicemos un navegador que no nos muestra la imagen o nos hable el texto alternativo.

El comportamiento del atributo `title` sí que indica que tiene que aparecer a modo de tooltip. Resumiendo, Internet Explorer no interpretaba correctamente el comportamiento del atributo `alt` y el resto de los navegadores, sí. Por lo tanto, no debemos utilizar el atributo `title` para que en el resto de navegadores aparezca el tooltip.

Imágenes con enlace:

Es posible utilizar una imagen como enlace, basta con utilizar la imagen en vez del texto del enlace:

Ejemplo:

```
<a href="index.html"></a>
```

Formato de imagen a utilizar:

Los formatos a utilizar son los generales para imágenes: PNG, JPG y GIF, SVG y WEBP.

Mapas de Imágenes:

Un mapa de imágenes es simplemente una imagen sobre la que se definen unas áreas activas (hotspots) que actuarán como enlaces. Si tomamos una imagen de un mapa del mundo, por ejemplo, cada continente podría ser un área activa.

Existen dos tipos de mapa de imagen: basados en servidor (muy poco frecuente) y en cliente. En este caso, la definición de las áreas activas del mapa se hace dentro del html, por lo que es funcional incluso sin conexión ni carga de imágenes.

La imagen que se convertirá en un mapa de imagen, primero debemos incluirla en nuestra página (con el elemento ``) y después debemos indicar que se trata de un mapa de imágenes mediante el atributo `usemap` con el valor igual al del nombre que se le vaya a dar al mapa de imagen. El nombre debe estar precedido del signo #.

Ahora ya podemos pasar a definir el mapa de imagen. Utilizaremos el elemento `<map>` y lo identificaremos mediante el atributo `id` con el valor igual al del nombre que le hemos puesto en la llamada anterior.

Con el elemento `<area>` podemos definir las áreas activas. Con el atributo `shape`, determinamos la forma de esa área, que, dependiendo del valor asociado, podrá ser cuadrangular, circular o poligonal. Finalmente, con el atributo `coords`, especificaremos las coordenadas de la superficie de esa área activa. Normalmente, estas coordenadas se expresan en píxeles, separadas con comas.

El área cuadrangular (`shape="rect"`) se define con las coordenadas para `x1, y1, x2, y2`:

- `x1` = esquina superior izquierda, píxel de la izquierda
- `y1` = esquina superior izquierda, píxel de arriba
- `x2` = Esquina inferior derecha, píxel de la izquierda
- `y2` = Esquina inferior derecha, píxel de arriba

El área circular (`shape="circle"`) se define con las coordenadas para `x, y, r`:

- `x` = Punto central, píxel de la izquierda
- `y` = Punto central, píxel de arriba
- `r` = Radio en píxel

El área poligonal (`shape="polygon"`) se define con las coordenadas `x1, y1, x2, y2...`:

- `x` = Píxel de una esquina de la izquierda
- `y` = Píxel de una esquina de arriba

Se pueden definir tantas esquinas como queramos. Desde la última esquina nos imaginaremos una línea a la primera esquina. Esa esquina es la que cierra el polígono.

Para completar el mapa de imágenes, enlazaremos cada área activa con una URL (mediante el atributo `href`) y finalmente, añadiremos un texto alternativo para cada una de las áreas (mediante el atributo `alt`).

Ejemplo:

```

<map id="planetmap">
<area shape="rect" coords="0,0,82,12" href="sun.html" alt="Sun" />
<area shape="circle" coords="0,5,1" href="moon.html" alt="Moon" />
<area shape="circle" coords="1,0,8" href="mars.html" alt="Mars" />
</map>
```

4. TÍTULOS (HEADINGS):

Mediante los títulos, estructuraremos la información de la página en base a las secciones que tenga.

Disponemos de 6 elementos de título (de la h1 a la h6) donde h1 es el título más importante.

Lo normal es no utilizar un h sin haber utilizado previamente su h padre, es decir. No deberíamos utilizar h2 en una página/elemento si, previamente, no hemos utilizado h1.

Ejemplo:

```
<h1>Servicios</h1>
<p>Empresa SA le proporciona los siguientes servicios:</p>

<h2>Catering</h2>
<p>Por un módico precio, puede disfrutar de la mejor y más variada comida del mundo...</p>

<h2>Preparación oposiciones</h2>
<p>Nuestro personal estará encantado en ayudarle a preparar las siguientes oposiciones:</p>

<h3>Ayudante de Chef</h3>
<p>bla, bla, bla</p>

<h3>Ayudante de cocina</h3>
<p>bla, bla, bla</p>

<h2>Formación privada</h2>
<p>Si lo desea, podemos enseñarle a preparar usted mismo los exquisitos platos...</p>
```

Servicios

Empresa SA le proporciona los siguientes servicios:

Catering

Por un módico precio, puede disfrutar de la mejor y más variada comida del mundo...

Preparación oposiciones

Nuestro personal estará encantado en ayudarle a preparar las siguientes oposiciones:

Ayudante de Chef

bla, bla, bla

Ayudante de cocina

bla, bla, bla

Formación privada

Si lo desea, podemos enseñarle a preparar usted mismo los exquisitos platos....

Separadores horizontales:

Los separadores horizontales sirven para hacer una separación del contenido de la página.

Ejemplo:

```
<h1>Servicios</h1>
<p>Empresa SA le proporciona los siguientes servicios:</p>

<h2>Catering</h2>
<p>Por un módico precio, puede disfrutar de la mejor y más variada
comida del mundo...</p>

<hr />

<h2>Preparación oposiciones</h2>
<p>Nuestro personal estará encantado en ayudarle a preparar las
siguientes oposiciones.</p>
```

Servicios

Empresa SA le proporciona los siguientes servicios:

Catering

Por un módico precio, puede disfrutar de la mejor y más variada comida del mundo...

Preparación oposiciones

Nuestro personal estará encantado en ayudarle a preparar las siguientes oposiciones.

5. LISTAS:

Listas desordenadas:

Los elementos de la lista no tienen un orden determinado.

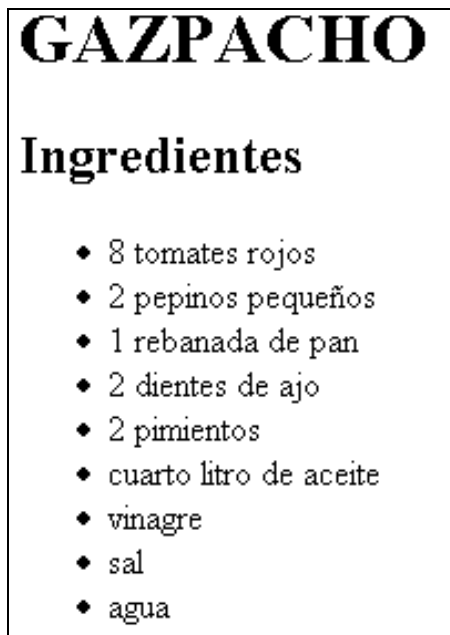
A nivel visual, cada elemento de la lista aparece con una viñeta, sin ninguna numeración. Se representan por el elemento `ul` (unordered list).

Ejemplo:

```
<h1>GAZPACHO</h1>

<h2>Ingredientes</h2>

<ul>
<li>8 tomates rojos</li>
<li>2 pepinos pequeños</li>
<li>1 rebanada de pan</li>
<li>2 dientes de ajo</li>
<li>2 pimientos</li>
<li>cuarto litro de aceite</li>
<li>vinagre</li>
<li>sal</li>
<li>agua</li>
</ul>
```



Listas ordenadas:

Los elementos de la lista tienen un orden específico.

Las listas ordenadas numeran sus elementos. Se representan por el elemento `ol` (ordered list):

Ejemplo:

```
<h1>GAZPACHO</h1>

<h2>Preparación</h2>
```

```

<ol>
<li>Triturar en la batidora los pimientos y los tomates partidos por la
mitad, los pepinos pelados, los ajos, el pan, y el aceite, con un
chorrito de vinagre, un poco de agua y sal.</li>
<li>Batir muy bien</li>
<li>Vertir todo en la fuente en la que vaya a ser servido y agregar
agua hasta que adquiera la consistencia de una crema ligera.</li>
<li>Servir muy frío.</li>
</ol>

```

GAZPACHO

Preparación

1. Triturar en la batidora los pimientos y los tomates partidos por la mitad, los pepinos pelados, los ajos, el pan, y el aceite, con un chorrito de vinagre, un poco de agua y sal.
2. Batir muy bien
3. Vertir todo en la fuente en la que vaya a ser servido y agregar agua hasta que adquiera la consistencia de una crema ligera.
4. Servir muy frío.

Gracias a CSS podemos modificar la apariencia de las listas: Cambiar las viñetas, el tipo de numeración (números romanos, letras...) y la orientación de las listas (vertical u horizontal).

Listas anidadas:

Debemos tener mucho cuidado al etiquetar las listas anidadas, puesto que la mayor parte de editores de código no lo hacen correctamente y por lo tanto, no cumplen la especificación XHTML. Tendremos que revisarlo manualmente:

Correcto:

```

<ul>
<li>Trilogía de El señor de los anillos
<ol>
<li>La comunidad del anillo</li>
<li>Las dos torres</li>
<li>El retorno del rey</li>
</ol>
</li>
</ul>

```

LIBROS MÁS VENDIDOS

- ♦ Trilogía de El señor de los anillos
 1. La comunidad del anillo
 2. Las dos torres
 3. El retorno del rey

Incorrecto:

```
<ul>
<li>Trilogía de El señor de los anillos</li>
  <ol>
    <li>La comunidad del anillo</li>
    <li>Las dos torres</li>
    <li>El retorno del rey</li>
  </ol>
</ul>
```

Listas de definición:

A diferencia de las listas anteriores, las listas de definición, representadas por el elemento `dl`, tienen un par de elementos por cada ítem: un término, `dt`, y su definición, `dd`. Es posible que un término pueda tener varias definiciones:

Ejemplo:

```
<h1>Ficha técnica</h1>

<dl>
  <dt>Nombre:</dt>
  <dd>Open open</dd>

  <dt>Versión:</dt>
  <dd>2.0</dd>

  <dt>Plataformas:</dt>
  <dd>Linux</dd>
  <dd>Mac</dd>
  <dd>Windows</dd>
</dl>
```

Ficha técnica	
Nombre:	Open open
Versión:	2.0
Plataformas:	Linux Mac Windows

Uso de listas en menús:

Siguiendo con esta lógica, cuando tengamos que implementar un menú en nuestra página web, lo haremos mediante listas.

Ejemplo:

```
<ul>
  <li><a href="index.html">Home</a></li>
  <li><a href="contct.html">Contact</a></li>
  <li><a href="webmap.html">Webmap</a></li>
  <li><a href="access.html">Accessibility</a></li>
</ul>
```

[Home](#) : [Contact](#) : [Webmap](#) : [Accessibility](#)

[Home](#) > [News & Events](#) > [Notice Board](#)

Y como apreciamos, también las utilizaremos para el camino de migas de pan, la paginación de unos resultados, etc... Bastará con que utilicemos CSS para cambiar el aspecto gráfico⁴. Los límites los pone nuestra imaginación.

⁴ Puedes revisar distintos estilos para las listas en:
<http://css.maxdesign.com.au/listamatic/>

6. TABLAS:

Las tablas sirven para presentar información de forma tabulada. Hasta hace unos años, se utilizaban erróneamente para maquetar una página, a modo de rejilla, por lo que su valor semántico era incorrecto.

Tablas básicas:

Los elementos básicos para construir una tabla son: `<table>`, `<thead>` `<tfoot>` `<tbody>` `<tr>`, `<th>` y `<td>`. También podemos utilizar `<caption>` si queremos poner un título a la tabla.

Ejemplo:

```
<table> <!-- comienza la tabla -->

<!-- título de la tabla -->
<caption>Personal seleccionado para la fase final</caption>

<thead>
  <tr> <!-- empezamos a escribir la 1ª fila -->
    <!-- escribimos las celdas de cabecera -->
    <th>Nombre</th>
    <th>Apellido</th>
    <th>Edad</th>
  </tr> <!-- finalizamos la 1ª fila -->
</thead>

<tbody>
  <tr> <!-- empezamos a escribir la 2ª fila -->
    <!-- escribimos las celdas de datos -->
    <td>Ander</td>
    <td>Téllez</td>
    <td>23</td>
  </tr> <!-- finalizamos la 2ª fila -->

  <tr> <!-- empezamos a escribir la 3 fila -->
    <!-- escribimos las celdas de datos -->
    <td>Andoni</td>
    <td>Merino</td>
    <td>25</td>
  </tr> <!-- finalizamos la 3 fila -->
</tbody>

</table> <!-- finaliza la tabla -->
```

Personal seleccionado para la fase final		
Nombre	Apellido	Edad
Ander	Téllez	23
Andoni	Merino	25

Expansión de filas y columnas.

En ocasiones, es necesario que una celda ocupe más de un espacio, para ello, utilizaremos los atributos `colspan` y `rowspan`.

Ejemplo:

```

<table>
<caption>Novedades en el Videoclub</caption>
<tr>
  <th>Título</th>
  <th colspan="2">Género</th>
</tr>

<tr>
  <td>Sin City</td>
  <td>Acción</td>
  <td>Comic</td>
</tr>

<tr>
  <td>Charlie y la fábrica de chocolate</td>
  <td>Infantil</td>
  <td>Musical</td>
</tr>

<tr>
  <td>La madre del novio</td>
  <td rowspan="2">Comedia</td>
  <td rowspan="2">Romántica</td>
</tr>

<tr>
  <td>Embrujada</td>
</tr>

</table>

```

Novedades en el Videoclub		
Título	Género	
Sin City	Acción	Comic
Charlie y la fábrica de chocolate	Infantil	Musical
La madre del novio	Comedia	Romántica
Embrujada		

Para modificar las propiedades de bordes, espacio entre celdas, márgenes dentro de la celda y aspecto gráfico, utilizaremos las propiedades de CSS, p.e.

Lista de todos los enlaces insertados (6 en total)			
ID Enlace	Fecha	Categoría	Enlace
 Enlace 19	01/04/2005 2:28:30	otras asociaciones	Expolingua 2005
 Enlace 17	09/03/2005 17:45:15	foros y listas	Listas de lengua, gramática y literatura
 Enlace 16	09/03/2005 17:37:47	foros y listas	Listas de correo para traductores
 Enlace 13	07/03/2005 12:08:47	otras asociaciones	Atic
 Enlace 10	07/03/2005 11:53:21	foros y listas	Translator Client Review
 Enlace 9	07/03/2005 11:49:44	foros y listas	Payment practices

7. FORMULARIOS

Utilizamos los formularios para poder recoger la información que los usuarios introducen en las páginas web. Después la procesaremos (preferiblemente en servidor) y guardaremos la información en una base de datos, o generaremos una página dinámica...

Primero vamos a explicar el funcionamiento y campos básicos de un formulario y más adelante veremos campos avanzados.

Formulario:

Para implementar un formulario utilizaremos el elemento `<form>`. Dentro de este elemento ubicaremos los campos de formulario.

Además, un formulario necesita una acción (lo que se ejecutará cuando se envíen los datos), un método (método de transmisión de los datos) y una codificación para los datos (formato en que se van a enviar).

Existen 2 métodos de transmisión de datos: `get` (los datos se pasan por url) y `post` (los datos se guardan en "memoria"), el adecuado para enviarlos con un mailto.

Para la codificación de los datos podemos utilizar `text/plain` para enviar los datos como texto plano sin ningún formato (el adecuado para enviar datos por e-mail) o `multipart/form-data`, el formato adecuado para enviar ficheros adjuntos.

Ejemplo:

```
<form action="mailto:info@empresa.com" method="post"
  enctype="text/plain">
... campos del formulario ...
</form>
```

En este ejemplo hemos indicado que los datos se van a enviar a una dirección de correo electrónico (`action`), enviando los datos por memoria (`method="post"`) y codificados en texto plano

Ejemplo:

```
<form action="contact.php" method="get" enctype="text/plain">
... campos del formulario ...
</form>
```

En este otro ejemplo, en cambio, hemos indicado que los datos se van a procesar en una página (`contact.php`) enviando los datos por url (`method="get"`) y codificados en texto plano.

Siempre que podamos, es preferible utilizar `method="post"` puesto que el método `get` es muy inseguro. El usuario puede ver los parámetros en la barra de direcciones y podría manipular a mano los valores.

Identificación de campos:

Los campos de formulario se identifican con el atributo `id`. Esto implica que nada (ningún otro elemento) puede llamarse de la misma manera (es decir, tener un `id` con el mismo valor, un `id` repetido).

Un `id` puede llevar cualquier valor, siempre y cuando empiece por una letra [A-Z, a-z] y puede contener cualquier número de letras, dígitos [0-9], guiones (- _) y puntos (: ..). No puede contener espacios.

Mediante el atributo `name` damos nombre a la variable de cada campo. De esta manera, tendremos cómo recibir las variables al ejecutar un formulario.

Para evitarnos problemas a la hora de identificar las variables, es recomendable escribir el mismo valor para `id` y `name` (salvo en algunas excepciones que veremos más adelante).

Asociación de etiquetas de texto a campos de formulario:

Cuando insertamos un campo de formulario, debemos indicar qué es lo que se debe introducir en ese campo. Para ello escribimos una cadena de texto que precede al campo.

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

Normalmente, en un navegador gráfico, no tendremos dificultades de determinar que el texto Nombre es el que acompaña al campo de formulario que está después, pero en navegadores sólo-texto o con lectores de pantalla, no podemos asegurarlo. Por ello, debemos asociar el texto al campo de formulario.

Para realizar la asociación, utilizaremos el elemento `<label>`, que será la encargada de asociar texto con su campo de formulario correspondiente. En este elemento utilizaremos el atributo `for`, cuyo valor será el mismo de la `id` del campo al que asociamos:

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

De esta forma, estamos haciendo una asociación explícita de la etiqueta de texto con su campo de formulario (incluso en el código de nuestra página los elementos `<label>` e `<input>` no tendrían por qué estar juntos).

Nota Accesibilidad:

Cuando estamos implementando una página accesible es necesario asociar de forma implícita las etiquetas de texto con su campo de formulario correspondiente. La asociación implícita se puede hacer de dos formas:

Mediante englobamiento: Basta con englobar el elemento `<input>` con el elemento `<label>`:

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

Mediante posición: Es necesario que no haya ningún elemento entre `<label>` y el campo de formulario:

Incorrecto:

```
<label for="nombre">Nombre:</label>

<input type="text" name="nombre" id="nombre" />
```

Correcto:

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

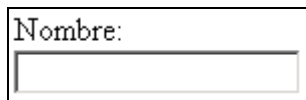

Campos de texto

Para la mayor parte de los campos de un formulario se utiliza el elemento `<input>`. Modificando su atributo `type` con distintos valores, podremos cambiar de tipo de campo.

Podemos indicar también el ancho del campo (medido en caracteres) mediante el atributo `size` y el número máximo de caracteres que se podrán escribir en el campo, con el atributo `maxlength`. También será posible establecer un valor por defecto en el campo, gracias al atributo `value`.

Para un campo de texto básico, utilizamos el atributo `type` con el valor `text`.

```
<p><label for="nombre">Nombre:</label><br />
<input type="text" id="nombre" name="nombre" size="20" /></p>
```

Una captura de pantalla de un formulario web. A la izquierda, el texto "Nombre:" actúa como etiqueta. A la derecha, hay un campo de entrada de texto rectangular con un borde gris, listo para recibir texto.

Campos de contraseña:

Estos campos son idénticos a los de texto, sólo que cuando el usuario escribe en ellos, en vez de aparecer letras, aparecen asteriscos. Para ello, utilizamos el atributo `type` con el valor `password`.

```
<p><label for="pass">Password: </label><br />
<input type="password" name="pass" id="pass" size="8" /></p>
```

Una captura de pantalla de un formulario web. A la izquierda, el texto "Password:" actúa como etiqueta. A la derecha, hay un campo de entrada de contraseña rectangular con un borde gris. El contenido del campo está oculto por una serie de caracteres de relleno (asteriscos o puntos).

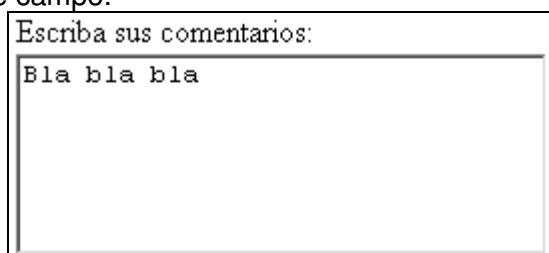
Áreas de texto:

Las áreas de texto se utilizan cuando en un campo, el usuario necesita introducir gran cantidad de texto, en varias líneas. Utilizamos el elemento `<textarea>`, que difiere al del elemento `<input>`. Este elemento sí que tiene etiqueta de cierre.

Para especificar el ancho del área de texto (medido en caracteres) utilizaremos el atributo `cols`, y para el número de líneas, `rows`.

```
<p><label for="comentarios">Escriba sus comentarios:</label><br />
<textarea name="comentarios" id="comentarios" cols="30" rows="5">
Bla bla bla
</textarea>
</p>
```

El texto que haya entre la etiqueta de apertura y la de cierre será el valor por defecto de este campo.

Una captura de pantalla de un formulario web. A la izquierda, el texto "Escriba sus comentarios:" actúa como etiqueta. A la derecha, hay un área de texto rectangular con un borde gris. El área de texto contiene el texto "Bla bla bla" en su primera línea, y las líneas restantes están vacías.

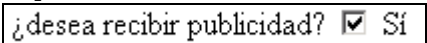
Checkbox o casillas de verificación:

Las checkbox se representan gráficamente como un cuadrado que el usuario puede activar y desactivar.

Para crearlos, utilizaremos el atributo `type` con el valor `checkbox`. En este caso, el atributo `value` funciona de forma distinta. El valor que tenga ese atributo será el valor que recogeremos al ejecutar el formulario, si esa casilla está activada.

Normalmente se coloca primero el campo checkbox y después su texto asociado, al contrario que con el resto de los campos:

```
<p>
¿desea recibir publicidad?
<input type="checkbox" name="publi" id="publi" value="si"
checked="checked" />
<label for="publi">Sí</label>
</p>
```



Si este ejemplo fuera parte de un formulario, enviado a una página que procese los datos por programación, recibiríamos `publi=si`

Radiobuttons o botones de selección:

Los radiobuttons se representan gráficamente por botones redondos. Se utilizan normalmente cuando ofrecemos al usuario una lista de opciones de la que sólo puede seleccionar una de ellas, de forma exclusiva.

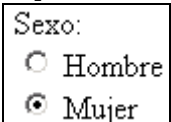
Este caso de elemento `<input>` es el más especial que hay. Utilizaremos el atributo `type` con el valor `radio`.

Imaginemos que en un cuestionario queremos preguntar el sexo de la persona. Las opciones son `hombre` y `mujer` y el usuario sólo va a poder elegir una de ellas.

Lo primero que tenemos que hacer es agrupar los 2 botones de opciones (y así crear un grupo de opciones). Para ello damos a cada elemento `input` el mismo valor en el atributo `name`. El atributo `value` deberá llevar el valor que queramos recoger en el formulario y el valor del atributo `id` podrá ser una combinación de ambos, por ejemplo:

En el caso de los radiobuttons, también se coloca primero el campo radiobutton y después su texto asociado, al igual que los checkbox:

```
<p>Sexo:<br />
<input type="radio" name="sexo" id="sexohombre" value="hombre" />
<label for="sexohombre">Hombre</label><br />
<input type="radio" name="sexo" id="sexomujer" value="mujer"
checked="checked" />
<label for="sexomujer">Mujer</label><br />
</p>
```



Si procesáramos este formulario y el usuario marcara la opción "hombre", recibiríamos `sexo=hombre`.

Nota Usabilidad:

Podríamos hacer que unos radiobuttons funcionaran como los checkbox, es decir, que se pudiera hacer una selección múltiple, simplemente con poner valores diferentes en sus atributos `name`. De la misma manera, podríamos hacer que los checkbox funcionaran como los radiobuttons, si ponemos un mismo valor en los atributos `name` de todos los checkbox (creando el grupo de opciones).

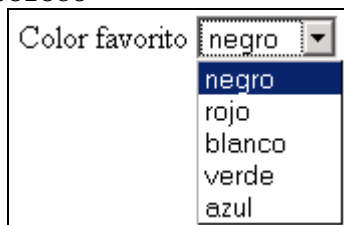
Estas técnicas están desaconsejadas porque el usuario ha aprendido a identificar los checkbox y los radiobuttons y a diferenciar su comportamiento, por lo que si lo modificamos, entorpeceríamos la interacción del usuario con la página y podríamos alterar su patrón de conocimiento.

Listas de selección desplegables:

Las listas de selección tienen una función similar a la de los radiobuttons. Presentamos al usuario múltiples opciones en las que, a priori, solo puede seleccionar una de ellas. La ventaja que tienen las listas desplegables es que apenas ocupan espacio, y son muy útiles cuando tenemos muchas opciones.

Utilizamos para implementarlas el elemento `<select>`. Por cada opción, tendremos que utilizar un elemento `<option>` con el atributo `value`, con el valor que deseemos recoger al procesar el formulario y el valor que queramos que aparezca por pantalla (si el valor por pantalla es igual que `value`, podemos prescindir de `value`).

```
<label for="color">Color favorito</label>
<select name="color" id="color">
  <option value="negro">negro</option>
  <option value="rojo">rojo</option>
  <option value="blanco">blanco</option>
  <option value="verde">verde</option>
  <option value="azul" selected="selected">azul</option>
</select>
```



Si procesáramos este formulario y el usuario marcara la opción "verde", recibiríamos `color=verde`.

Si tenemos un listado que queremos categorizar, podemos utilizar el elemento `optgroup` para agrupar las opciones de una determinada categoría y el atributo `label` (en este caso actúa como atributo y no como etiqueta) para indicar la categoría.

```
<label for="producto">Seleccione un producto:</label>
<select name="producto" id="producto">
  <optgroup label="software">
    <option value="1">Ubuntu</option>
    <option value="2">Gimp</option>
    <option value="3">Open Office</option>
  </optgroup>
  <optgroup label="hardware">
    <option value="4">Regrabador DVD</option>
    <option value="5">Tarjeta Gráfica</option>
  </optgroup>
  <optgroup label="material">
    <option value="7">Tarrina 10 CDs</option>
  </optgroup>
</select>
```

```

        <option value="8">Alfombrilla</option>
        <option value="9">WebCam</option>
    </optgroup>
</select>

```



Con estos elementos podemos olvidarnos del uso de elementos `option` con valor vacío para indicar la categoría y así evitarnos las comprobaciones de las opciones (no válidas) elegidas.

Botones:

Para poder enviar o restablecer (borrar) los datos del formulario, necesitamos botones. Ambos se implementan con el elemento `<input>`. Si utilizamos el atributo `type` con el valor `submit`, creamos el botón de envío; si, por el contrario, `type` lleva el valor `reset`, creamos el botón de borrado.

El atributo `name` no tendrá mucho sentido, salvo que empleemos algún `script` para tratar la información, e `id` tampoco, salvo que vayamos a aplicarle un `script` o `estilo css` exclusivo. El atributo `value` nos servirá para indicar el texto que queramos que aparezca en el botón.

```

<input type="submit" value="enviar datos" />
<input type="reset" value="borrar datos" />

```



El botón de envío se encarga de enviar la información del formulario (de la forma que lo hayamos determinado en el atributo `action` del elemento `<form>`) y el botón de reseteo, vuelve a poner los valores por defecto del formulario.

Existe otro tipo de botón, que no ejecuta nada por sí solo. Es necesario aplicarle una función `javascript` para accionarlo. El valor que debemos poner al atributo `type` es `button`:

```

<input type="button" value="calcular" onclick="calc();" />

```

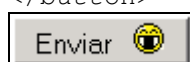
Botones complejos:

Podemos utilizar el elemento `<button>` para implementar botones más complejos y con mayor carga de contenido, como texto y/o imágenes:

```

<button name="boton" type="submit">
Enviar 
</button>

```



Botones gráficos:

Podemos utilizar imágenes que actúen como botones de tipo `submit`. Basta con implementar el elemento `<input>`, utilizar el valor `image` en el atributo `type`, indicar la ruta del archivo en el atributo `src` y determinar un texto alternativo a la imagen con el atributo `alt`.

```
<input type="image" src="boton.gif" alt="Enviar" />
```



Botones de archivo:

Si necesitamos recoger un archivo, en vez de un texto, necesitamos implementar el botón de tipo archivo. Para ello, utilizaremos el elemento `<input>` con el valor `file` en el atributo `type`:

```
<input type="file" name="ejem" id="ejem" />
```



Si especificamos el atributo `maxlength`, el navegador lo interpretará como el tamaño máximo permitido en bites

```
<input type="file" maxlength="100000" name="dato" id="dato" />
```

Podemos controlar el tipo de archivos admitidos mediante el atributo `accept` y distintos valores de tipo Mime. Con `accept="text/*"` aceptamos cualquier tipo de archivo.

```
<input type="file" name="ejem" id="ejem" accept="image/gif" />
```

Lo más importante es determinar la especificación `enctype="multipart/form-data"` en el elemento `<form>`, cuando el formulario contiene un botón de tipo archivo.

Elementos ocultos:

Podemos definir campos de formulario ocultos con el elemento `<input>` y el valor `hidden` en el atributo `type`.

```
<input type="hidden" name="codigo" id="codigo" value="a1">
```

Elementos deshabilitados y de sólo lectura:

Podemos definir campos de formulario deshabilitados, con el par de atributo/valor `disabled="disabled"` y de sólo lectura, mediante `readonly="readonly"`.

```
<input type="text" id="nombre" name="nombre" disabled="disabled" />
<input type="text" id="nombre" name="nombre" readonly="readonly" />
```

Si utilizamos el atributo `disabled`, el valor que tenga ese campo no se enviará con el formulario. Sin embargo, un campo con el atributo `readonly` sí.

Categorizar y agrupar campos similares o diferentes formularios:

En una página web nos podemos encontrar con tres casos donde podemos utilizar el elemento `fieldset`, y para indicar el título de cada bloque, utilizamos el elemento `legend`.

1) En un formulario tenemos datos de diferente tipo (p.e. datos personales y datos bancarios). Utilizaremos el elemento `fieldset` para agrupar cada tipo de dato:

```
<form...>
  <fieldset>
    <legend>Datos personales</legend>
    <label for="nombre">Nombre:</label>
    <input type="text" name="nombre" id="nombre" /><br />
    <label for="e-mail">E-mail</label>
    <input type="text" name="e-mail" id="e-mail" />
  </fieldset>

  <fieldset>
    <legend>Datos bancarios</legend>
    <label for="banco">Banco:</label>
    <input type="text" name="banco" id="banco" /><br />
    <label for="numcuenta">Número de cuenta:</label>
    <input type="text" name="numcuenta" id="numcuenta" />
  </fieldset>
</form>
```

Datos personales Nombre: <input type="text"/> E-mail: <input type="text"/>	Datos bancarios Banco: <input type="text"/> Número de cuenta: <input type="text"/>
-----------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------

2) En la página tenemos más de un formulario (p.e. formulario de contacto y formulario de búsqueda). Utilizaremos un elemento `fieldset` para agrupar cada uno de los formularios.

```
<fieldset>
<legend>Formulario de contacto</legend>
  <form...>
    Campos del formulario de contacto
  </form>
</fieldset>

<fieldset>
<legend>Búsqueda Avanzada</legend>
  <form...>
    Campos del formulario de buscador
  </form>
</fieldset>
```

3) En un formulario, tenemos un campo de selección de más datos (p.e. tenemos varios `radiobuttons` para seleccionar el género).

```
<form>
  <fieldset>
    <legend>Sexo:</legend>
    <input type="radio" name="sexo" id="sexohombre" value="hombre" />
    <label for="sexohombre">Hombre</label><br />
    <input type="radio" name="sexo" id="sexomujer" value="mujer" checked="checked" />
    <label for="sexomujer">Mujer</label><br />
  </fieldset>
</form>
```

8. ELEMENTOS META

Los elementos meta van ubicadas en el documento HTML dentro del elemento `<head>`. Proveen información sobre la página, fundamentalmente a los buscadores.

Existen elementos meta de 2 tipos distintos: elementos `http-equiv` y atributos `name`.

Elementos http-equiv:

Equivalen a las cabeceras HTTP, y las más utilizadas son:

Expires

Fecha y hora después de la cual la página se considera caducada. Las fechas deben escribirse en formato RFC850, en GMT:

```
<meta http-equiv="expires" content="Wed, 26 Feb 1997 08:21:57 GMT" />
```

Pragma

Se emplea para prevenir que los navegadores cacheen la página localmente.

```
<meta http-equiv="Pragma" content="no-cache" />
```

Refresh

Permite realizar un redireccionamiento de página a una URL alternativa o bien recargar la misma página después de un determinado tiempo.

```
<meta http-equiv="refresh" content="3;url=http://www.web.org" />
```

Atributos name:

Los elementos meta, con el atributo `name` se utilizan para otros tipos que no corresponden con las cabeceras HTTP. Los más utilizados son:

Robots

Se utiliza para modificar el comportamiento de los robots ante la página. Podemos indicar qué páginas puede indexar y cuáles no, así como las páginas por las que puede hacer un recorrido, mediante los valores siguientes:

- **index / noindex:** permite / evita el indexado de la página.
- **follow / nofollow:** permite / evita que el robot siga los enlaces de la página y los indexe.

En este ejemplo, indicaríamos que el robot no puede indexar la página, pero sí puede "atravesar" la página e indexar las páginas enlazadas:

```
<meta name="robots" content="noindex, follow" />
```

Description

Lo debemos utilizar para ofrecer una descripción corta del documento:

```
<meta name="description" content="manual de HTML" />
```

Keywords

Se utilizaba para ofrecer a los motores de búsqueda una serie de palabras para la correcta indexación de la página. Sin embargo, hace muchos años que Google ha dejado de utilizarlo.

```
<meta name="keywords" content="html, css, estandares web" />
```

Author

Normalmente se utiliza para indicar el autor de la página..

```
<meta name="author" content="Pepito Grillo" />
```

Generator

Generalmente se indica el nombre y la version de la herramienta de autor empleada en la creación de la página web.

```
<meta name="generator" content="notepad" />
```

Copyright

Se utiliza para indicar los derechos legales de la página.

```
<meta name="copyright" content="Pepito Grillo 2018" />
```


9. ELEMENTOS LINK

Estos elementos ofrecen información extra a los usuarios y navegadores que tienen soporte para interpretarlo de cómo se relaciona un documento con otros.

Stylesheet

Se refieren a una hoja de estilos externa. Si se usa conjuntamente con el enlace de tipo `Alternate` podemos ofrecer una hoja de estilos alternativa, seleccionable por los usuarios.

Alternate

Señala las versiones sustitutas para el documento. Si se utiliza conjuntamente con el atributo `lang`, implica una versión traducida del documento. Si se usa conjuntamente con el atributo `media`, implica una versión diseñada para un medio distinto.

Contents

Hace referencia a un documento que sirve como tabla de contenidos (mapa web, por ejemplo). Algunos agentes de usuario también soportan el sinónimo `ToC`.

Glossary

Hace referencia a un documento que sirve como glosario de términos perteneciente al documento actual.

Copyright

Hace referencia a una declaración del copyright para el documento actual.

Appendix

Hace referencia a un documento que sirve como apéndice en una colección de documentos.

Help

Hace referencia a una ayuda de ofrecimiento del documento (más información, enlaces a otras fuentes de información...)

Start

Hace referencia al primer documento en una colección de documentos. Este tipo de enlace sirve a motores de búsqueda para que identifiquen qué documento es considerado por el autor el punto de partida de la colección.

Next

Hace referencia al documento siguiente en una secuencia lineal de documentos. Los agentes del usuario pueden elegir cargar el documento `Next`, para reducir el tiempo de carga percibido.

Prev

Hace referencia al documento anterior en una serie pedida de documentos. Algunos agentes del usuario también apoyan el sinónimo `Previous`.

Index

Hace referencia a un documento que proporciona un índice para el documento actual.

Chapter

Hace referencia a un documento como capítulo en una colección de documentos.

Es posible añadir distintos tipos de link, teniéndolos que añadir en una especificación aparte. Hay casos, como en los del icono de acceso directo, que está ampliamente soportado por los navegadores actuales.

Icon:

Se utiliza para añadir un icono en la barra de navegación, junto a la dirección web. Normamente, si se añade esta página a los marcadores (bookmarks) o favoritos, aparecerá la imagen junto al nombre:

```
<link rel="icon" href="favicon.ico" type="image/x-icon">
```

A día de hoy, es posible indicar más iconos (optimizados para teléfonos móviles, diferentes resoluciones, etc)⁵

Otros:

Existen otras etiquetas META que se han ido creando por parte de diferentes redes sociales y otros servicios. Estas etiquetas no pertenecen de forma oficial a la especificación HTML (al ser etiquetas propias) pero se pueden utilizar sin problema.⁶

⁵ <https://github.com/audreyr/favicon-cheat-sheet>
<https://bitsofco.de/all-about-favicons-and-touch-icons/>

⁶ <https://getthead.info/>