Tema 2. HTML y XHTML

Tema 2. HTML y XHTML

1. Introducción.

HTML es el acrónimo de HyperText Markup Language. El significado de XHTML es similar, pero va precedido por "eXtensible", lo que nos indica que cumple las normas de XML. En este tema vamos a aprender HTML 4.01 y con tan solo asegurarnos de cumplir unos pequeños detalles su equivalente XHTML 1.0. También veremos algunas de las características más usadas de HTML 5.

HTML es un lenguaje basado en etiquetas que sirven para estructurar o dar formato a los contenidos de las páginas web. Junto al formato, es importante añadir conceptos de diseño o estilo, para lo cual, utilizamos lenguaje CSS (*Cascade Style Sheet*) que veremos en el siguiente tema.

En sus inicios HTML fue rápidamente acogido y empezó a crecer de una forma descontrolada con etiquetas nuevas y funcionalidades propias de cada navegador. Debido a esta falta de estandarización se propone utilizar como base XML para imponer reglas, pero usando las etiquetas de HTML Así nació XMHTL.

La diferencia principal entre XHTML y HTML es que XHTML al ser un derivado de XML genera documentos **bien formados**, los cuales cumplen ciertas reglas:

a) Cada elemento debe contar con su marca de apertura y de cierre, y la anidación de elementos debe ser correcta.

b) Debe existir un único elemento raíz denominado <html>.

c) Se debe incluir una instrucción de procesamiento XML que indique la versión de XML y la codificación de caracteres usada. <?xml version="1.0" encoding="UTF-8"?>

d) Debe existir una declaración <!DOCTYPE> para poder validar el documento frente a una DTD. (Las declaraciones de tipo de documento las veremos en el punto siguiente).

e) Las etiquetas siempre se escribirán en minúscula.

f) Los atributos se escriben con minúscula, seguidos de un signo igual y de un valor delimitado por comillas. Por ejemplo, style="valor".

g) Los elementos vacíos que en HTML son correctos, como
, en XHTML deben escribirse como
 o como
</br>.

Con estas normas podremos asegurar que un documento HTML 4.01 es XHTML. Es muy importante comprender que las normas que definen cómo se deben interpretar las etiquetas que veremos en este tema, así como, los estilos del siguiente, están redactadas por un comité, pero que la interpretación ha correspondido a muchas empresas de software, tantas como navegadores hay. Esto supone que, debido a que el lenguaje no es absolutamente preciso, a veces, dicha interpretación varía, provocando que la forma de visualizar una misma página web no sea igual con un navegador que con otro, incluso puede que no se vea correctamente una página con algunos navegadores. Además, de todo esto, debemos añadir que, sobre todo, *Internet Explorer* suele añadir funcionalidades o etiquetas particulares para sus navegadores.

Es absolutamente necesario probar las páginas web que confeccionemos en, al menos, los navegadores más utilizados para asegurarnos de su correcta visualización por la mayoría de los usuarios.

1.1. Declaración <!DOCTYPE>

Para que los navegadores conozcan qué versión de HTML o XHTML es utilizada en el documento que deben presentar, es necesario incluir una cabecera <!DOCTYPE> que lo especifique.

La declaración <!DOCTYPE>, indica el tipo de documento, la versión de HTML/XHTML que se usa y la DTD (Definición de Tipo de Documento).

Todo documento de un lenguaje de marcas tiene en común una gramática que define las marcas permitidas en dicho lenguaje, las que son obligatorias y cómo deben usarse. Una manera de definir dicha gramática es mediante una DTD.

Así pues, en una DTD se identifica la estructura de un documento mediante reglas que indican los nombres de los elementos, su significado, dónde pueden ser utilizados, qué pueden contener, etc. De esta manera los navegadores, al conocer la DTD del documento, pueden representarlo adecuadamente.

La etiqueta <!DOCTYPE> es la primera que se escribe en un documento HTML/XHTML, ya que va delante de la etiqueta <html>. Las DTD disponibles en las *Recomendaciones* del *W3C* para los documentos HTML y XHTML estandarizados actualmente son:

1.1.1. HTML 4.01 Strict

Esta DTD contiene todos los elementos y atributos HTML, pero no incluye elementos de presentación (estilos) o en desuso (como las etiquetas para fuentes). Los marcos tampoco están permitidos.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

1.1.2. HTML 4.01 Transitional

Esta DTD contiene todos los elementos HTML y atributos, incluyendo elementos de presentación y obsoletos. Los marcos no están permitidos. Es la versión HTML más usada.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

1.1.3. HTML 4.01 Frameset

Esta DTD es igual a HTML 4.01 Transitional, pero permite la utilización de marcos. De poco uso.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
```

1.1.4. XHTML 1.0 Strict

Esta DTD contiene todos los elementos y atributos HTML, pero no incluye elementos de presentación (estilos) o en desuso (como las fuentes). Los marcos no están permitidos. Además, el documento se debe escribir como XML **bien formado**.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

La etiqueta raíz <html> debe usarse de este modo (especificando el espacio de nombres y el lenguaje de la página):

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
```

1.1.5. XHTML 1.0 Transitional

Esta DTD contiene todos los elementos HTML y atributos, incluyendo elementos de presentación y obsoletos (como las fuentes). Los marcos no están permitidos. Además, el documento se debe escribir como XML **bien formado**. La etiqueta raíz <html> debe usarse como en el anterior. Es la versión XHTML más usada.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

1.1.6. XHTML 1.0 Frameset

Esta DTD es igual a XHTML 1.0 Transitional, pero permite la utilización de marcos. La etiqueta raíz <html> debe usarse como en el caso anterior.

```
<!DOCTYPE html>
```

1.1.7. XHTML 1.1

Esta DTD es igual a XHTML 1.0 Strict, pero se permite añadir ciertos módulos (por ejemplo, para proporcionar soporte para los idiomas de Asia Oriental). La etiqueta raíz <html> debe usarse como en el anterior.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

1.1.8. HTML 5

La especificación HTML 5 no está basada en SGML y, por lo tanto, no necesita la referencia a la DTD. Sin embargo, se requiere la declaración:

```
<!DOCTYPE HTML>
<html lang="es">
```

La etiqueta raíz HTML también es más sencilla al no indicar un espacio de nombres.

Nota: En http://www.webstandards.org/learn/reference/templates/ podemos encontrar plantillas para los diferentes tipos de documentos.

2. Estructura de una página web.

La estructura básica de una página web cambia según sea XHTML o HTML. Como son tantas las posibilidades, hablaremos de la versión más utilizada actualmente: HTML5. La estructura de una página HTML5 que esté escrita en español es:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset=UTF-8" />
        <title>Título</title>
    </head>
    <body>
        Contenido de la página web
    </body>
```

```
</html>
```

- La etiqueta **DOCTYPE** permite especificar el documento DTD de validación del HTML de la página. De esta forma, los clientes web conocerán sin ambigüedad con qué reglas deben procesar el documento. En caso de elegir otra versión de HTML, simplemente, hay que indicar las rutas al documento validador.
- Elemento **html** marca la raíz de la página indicando que todo el texto contenido entre <html> y </html> es código HTML. Por tanto, la primera debe abrir este tipo de documentos y la segunda, cerrarlos. Además, en versiones diferentes a HTML5, se debe especificar el espacio de nombres al que pertenecen las etiquetas (atributo **xmlns**) y el lenguaje en el que está escrito (en XHTML, atributo **xml:lang**).
- Es conveniente utilizar el atributo **lang**, que es común a todos los elementos HTML, y que sirve también para marcar el lenguaje en el que está escrita la página. Usando tanto **xml:lang** como **lang** aseguramos que todos los navegadores queden avisados sobre el lenguaje utilizado.

  El lenguaje español se puede indicar, simplemente, con **es** o, con más detalle, usando **es-ES**, que es el símbolo internacional de español de España. Esta terminología sigue los códigos de dos letras definidos en la norma ISO 639-1 (que contiene dos letras para cada lenguaje) y el código de dos letras de país de la ISO 3166-1.
- Elemento **head** marca el inicio de la cabecera. En la cabecera se coloca el título de la página (elemento **title**) y los elementos que sirven para incluir estilos CSS, código JavaScript y todo tipo de elementos que se utilizarán dentro de la página.
- La etiqueta **meta** advierte sobre el sistema de codificación del texto de la misma. En el código anterior se utiliza el más recomendable, que es UTF-8.
- Elemento **body** contiene el cuerpo de la página, es decir, el contenido visible por el navegador.

Las diferencias más importantes respecto a XHTML son:

- La etiqueta **DOCTYPE** está simplificada
- La etiqueta **meta** indica de forma mucho más sencilla el código del archivo Unicode.
- El lenguaje se especifica sólo con el atributo **lang** del elemento **html**.

2.1. Cabecera.

En la cabecera de un documento HTML se incluyen las etiquetas que dan información de la página o soporte para ella, pero que no se presentan como contenido de la página:

a) **<base />** Indica la dirección raíz del sitio web, la cual permite resolver las direcciones relativas. Si no se utiliza, los enlaces relativos parten del directorio que alberga a la página, de esta forma podemos hacer que los enlaces partan del directorio que digamos. Por ejemplo:

```
<base href ="http://www.paquito.es"/>
```

De tal forma que si tenemos una dirección relativa como "./imagenes/foto1.jpg", se completará dando lugar a "http://www.paquito.es/imagenes/foto1.jpg".

b) **<link />** Sirve para relacionar un documento con los recursos externos que utiliza como, por ejemplo, la hoja de estilos aplicable al documento. Se usan los atributos rel para indicar la relación del recurso con la página y href para indicar la URL del recurso. Otro atributo es type para indicar el tipo MIME del recurso. (Lista de tipos MIME en http://www.htmlquick.com/es/reference/mime-types.html). Ejemplos:

```
<link rel ="stylesheet" type="text/css" href="estilo.css"/>

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

Los posibles valores para el atributo rel son muchos, pero los más usados son:

- stylesheet. Es la que más se utiliza, indica que el recurso es una hoja de estilos (CSS).
- icon. Indica que el recurso es el icono de la página web. Se suele usar icon para iconos de tamaño 60x60, mientras que se usa short icon para iconos de tamaño 12x12 (normalmente se usan ambas entradas, escribiendo una etiqueta link distinta para cada tipo de icono).

c) **<title>** Es obligatorio y sirve para indicar el título de la página. Además, especifica el texto que aparecerá en la barra de título de la ventana. Es importante no confundir con que al hacer una página web la información que presentemos lleve un título, el cual deberemos poner con una etiqueta de encabezado <hn> antes de los párrafos de información.

d) **<meta />** Indica un conjunto de propiedades generales del documento que serán usadas por otro software, a la vez que proporciona información al navegador sobre la página. Se escriben tantas etiquetas meta, como informaciones se quieren registrar. Es frecuente que lleve los atributos:

- name. Indica el nombre de una propiedad. No hay una lista oficial de propiedades y, además, se crean nuevas con relativa frecuencia. Las más usuales son keywords, autor, description, copyright, date, expires, generator, organization, rating, robots, etc.
- content. Contiene el valor de la propiedad indicada por el atributo name.

Estos dos atributos se usan simultáneamente formando parejas. Por ejemplo:

```
<meta name="author" content="Paquito" />

<meta name="keywords" lang="es" content="instituto, zaidín, clase, ciclo" />

<meta name="description" lang="es" content="Ejemplos para la explicación" />

<meta name="robots" lang="es" content="index, nofollow" />
```

- http-equiv. Indica una propiedad al navegador en forma de cabecera HTTP como si el propio servidor web la hubiera generado. Se utiliza con el atributo content para proporcionar parejas de valores. Los valores posibles son: content-type, content-language, cache-control, set-cookie, refresh, resource-type, last-modified, etc. Por ejemplo:

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />

<meta http-equiv="content-language" content="es_Es" />

<meta http-equiv="refresh" content="10" />
```

- charset. Disponible desde HTML 5, sirve sólo para indicar la forma en la que está codificado el texto, en sustitución de la versión larga mediante http-equip="content-Type" de HTML 4 y XHTML. Ejemplo:

```
<meta charset="UTF-8" />
```

e) **<script>** Con esta etiqueta podemos incluir código en algún lenguaje cliente, por ejemplo, *JavaScript*. Esta etiqueta también puede aparecer una o más veces en la sección body. Se pueden usar los atributos:

- type. Indica el tipo MIME del contenido que se colocará dentro de la etiqueta. El habitual es text/javascript y es el que se asume por defecto.
- charset. Codificación usada para el texto del script. Por defecto toma el de la página.

- src. Permite indicar la URL de un archivo externo que contendrá el código (normalmente en *JavaScript*) que se colocará en la página.

Por ejemplo:

```
<script type="text/javascript">
    alert("hola")
</script>
```

f) **<style>.** Permite colocar código CSS en la página web. Se usa un atributo type para indicar el tipo de contenido que, casi siempre, será text/css.

```
<style type="text/css">
    body {background: black; color:white;}
    div {background: red; width:300px;}
</style>
```

No debemos olvidar que, además, es posible asignar estilos a una etiqueta determinada mediante su propio atributo style.

2.2. Estructura en zonas.

Para maquetar o dar una estructura a una página web, podemos dividirla en diferentes zonas lógicas a las que aplicaremos distintas posibilidades de diseño. Las zonas se definen con las siguientes etiquetas:

a) permite identificar un elemento en línea y, de esta forma, asignarle nombre, clase y estilo. Normalmente se usa para marcar contenido dentro de un párrafo, a fin de que a ese contenido se le pueda dar un formato especial mediante CSS

b) permite agrupar varios elementos dentro de un bloque y, así, asignarle nombre, clase y estilo. Normalmente se utiliza para definir capas en las páginas web.

Sus respectivas declaraciones son:

```
<span id="identificador" style="... ;... ;...; ...;">... </span>
<div id="identificador" style="... ;... ;...; ...;"> ... </div>
```

El atributo id permite poner un nombre o identificador a la zona o capa.

El atributo style, se utiliza para incluir una serie de propiedades de estilo. (Los estilos los veremos en profundidad en el siguiente tema, pero para entender mejor los div vemos un pequeño adelanto). Algunas propiedades básicas de estilo son:

- position: absolute | relative, indica si la posición de la zona es absoluta o relativa.
- top, indica el número de píxeles a desplazar desde la parte superior.
- left, indica el número de píxeles a desplazar desde la izquierda.

En el caso que la posición del elemento sea absoluta, los desplazamientos se miden desde los bordes del documento.

Por ejemplo:

```
<div id="nombre" style="position: absolute; top: 200px; left: 200px">  ...  </div>
```

Además, se pueden aplicar otras propiedades como background-color (indica el color de fondo), width (anchura de la zona), height (altura de la zona), text-align (alineación del texto), etc.

Ejercicio: Escribir el siguiente ejemplo en el *Bloc de notas* y llamarlo prueba1.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset=UTF-8" />
        <title>prueba1</title>
    </head>
    <body>
        <h1>Vamos a poner una etiqueta span y una div</h1>
        <span style="color:red; ">Esto está en una etiqueta span. No provoca salto de línea</span> fuera del
        span sigo en la misma línea y ya no soy de color rojo<br/>
        <div style="background-color: orange; border: 3px solid #C00; position: absolute; top: 200px">
            Este texto está dentro del div.
        </div>
    </body>
</html>
```

Una de las finalidades de HTML5 es que los elementos HTML sirvan para dar valor semántico al contenido más que para formatearlo. Es decir, para indicar qué tipo de contenido es. Lo que se pretende es indicar la semántica con HTML y dejar el formato para las hojas de estilo CSS.

En este sentido HTML5 incorpora etiquetas que no dan ningún formato al texto, pero permiten remarcarlo dándole un significado y posteriormente a ese contenido se le dará un formato especial mediante CSS.

Uno de los problemas del HTML "antiguo" es que no existían etiquetas capaces de agrupar bloques enteros y que tuviesen significado por sí mismas. Por eso, muchos desarrolladores terminaron usando como etiqueta para todo. HTML5 nos aporta una serie de etiquetas nuevas que permiten mejorar la semántica de nuestra página:<header>, <footer>, <nav>, <hgroup>, <aside>, <section>, <article>.

Esto no quiere decir que la etiqueta deje de usarse, sigue ahí y significando lo que siempre ha significado: conjunto de elementos. Se debe seguir usando como ayuda a la estructura de página para crear el *layout* de éstas, siempre y cuando no exista otra etiqueta de conjunto que pueda realizar este papel... Por lo tanto, es la herramienta para unir elementos cuando no podemos asociar significado semántico a este conjunto.

Un típico *layout* de una página web en HTML4 podría sustituirse por uno equivalente en HTML5 como se muestra a continuación:

```
<body>
    <div id="cabecera"> . . . </div>
    <div id="barra_navegacion">. . . </div>
    <div id="contenido">
        <div class="articulo">. . . </div>
        <div class="articulo">. . . </div>
```

```
            <div class="articulo">. . . </div>
            . . .
        </div>
        <div id="pie">. . . </div>
    </body>

    <body>
        <header> . . . </ header >
        <nav>. . . </nav>
        <section id="contenido">
            <article> . . . </article>
            <article> . . . </article>
            <article> . . . </article>
            . . .
            <aside> . . . </aside>
        </section>
        <footer>. . . </footer>
    </body>
```

El significado de estos nuevos elementos es:

- <header>. Permite marcar una cabecera para introducir información introductoria como son el título o el logo. Realmente una página puede tener varios elementos <header>. Al nivel de la etiqueta <body> indica que su contenido es la cabecera de la página. Pero dentro de una etiqueta, por ejemplo, <article> indicaría que su contenido es la cabecera del artículo.
- <footer> Sirve para marcar el pie de una página, sección, artículo, etc. Dependiendo del contexto en el que se coloque servirá para unas cosas u otras. Si se coloca al nivel del elemento <body>, servirá para agrupar los elementos de pie de página: el autor, copyright, términos de uso de la página, etc.
- <nav> Marca su contenido como una sección de enlaces, como por ejemplo una barra de navegación. Más adelante con CSS se puede dar un formato especial a dichos enlaces. <nav> se puede escribir dentro de cualquier elemento HTML de sección (como <section>, <article>, <header>, <footer>, ...).
- <hgroup> Permite agrupar varios elementos <h1> a <h6> para darles estilo común. Si se utiliza dentro de <article> permite marcar su zona de títulos.
- <aside> Permite indicar que ese bloque es solo un añadido a los bloques que tiene al lado. Son datos extra sin los que podríamos pasar perfectamente pero que hemos decidido añadir en el documento. Así, un <aside> como hijo de la etiqueta <body> nos dice que se trata de un contenido añadido por temas que no tienen nada que ver con el contenido de página (normalmente, esas columnas laterales llenas de *banners*). Si lo incluimos dentro de un <article> nos indica que esa información complementa el artículo, pero no forma parte de él (listas de datos, testimonios, banners relacionados, etc.).

- <section> Es un elemento que permite dividir en diferentes partes o secciones un documento. Puede anidarse para crear subsecciones. Englobando distintos elementos dentro de una etiqueta <section> lo que estamos haciendo es declarar que todo su contenido está relacionado y forma parte de un mismo significado o elemento.
- <article> La etiqueta pretende dar a entender que ese conjunto tiene significado claro incluso si lo sacamos totalmente de la página, como por ejemplo un post de un foro, un artículo periodístico, un comentario de usuario, etc. Hay que tener en cuenta que al ser un contenido con significado propio podría contener en su interior etiquetas <header>, <section>, <aside> y <footer>.

Vamos a mostrar un ejemplo práctico de uso de etiquetas contenedoras. Pensemos en una página con las siguientes partes:

- Cabecera con un logo y un menú de navegación.
- Debajo, el contenido a leer, con una serie de datos de referencia sobre este contenido.
- Finalmente, dos bloques en dos columnas con contenidos de forma libre.

Nuestra maqueta en HTML5 podría ser de la siguiente manera:

```
<body>
    <header>
        <p class="logo"><a href="...">...</a></p>
        <nav>
            <ul>
                <li><a href ... </li>
                <li><a href ... </li>
                . . .
            </ul>
        </nav>
    </header>
    <article>
        <header>
            <hgroup><h1>Mi título</h1><h3>Mi subtítulo</h3></hgroup>
        </header>
        <section>
            <p>Que si patatín, que si patatán …</p>
        </section>
        <aside>
            <p>Datos adicionales para completar el contenido</p>
        </aside>
    </article>
    <div class="bloques">
```

```
        <section>
            <hgroup><h2>Bloque 1</h2></hgroup>
            <ul>
                <li>...</li>
            </ul>
        </section>
        <section>
            <hgroup><h2>Bloque 2</h2></hgroup>
            <ul>
                <li>...</li>
            </ul>
        </section>
    </div>
    <footer>
        <ul>
            <li><a href="... [links del footer]</li>
        </ul>
    </footer>
</body>
```

3. Formato de etiquetas.

En HTML existen muchas etiquetas para marcar los elementos de una página, aunque algunas de ellas se consideran obsoletas, como las referentes a los marcos y a las fuentes o tipos de letra.

El formato general de una etiqueta es:

```
<etiqueta atributo="valor" atributo="valor"…> … </etiqueta>
```

Los atributos se pueden clasificar en:

a) **Atributos básicos**. Se usan en casi todas las etiquetas pero son realmente útiles para trabajar con hojas de estilo y/o scripts. Son los siguientes:

- id, identifica de forma única cada elemento de un documento HTML. Debe empezar por letra, no tener espacios, sí puede tener números y no puede repetirse su valor en dos elementos de la misma página web.
- class, determina clases de etiquetas para las hojas de estilo.
- style, establece, de forma directa, propiedades de estilo CSS a un elemento concreto.
- title, establece el título de un elemento que se muestra en una ventana emergente cuando se pasa el ratón por encima de él.

b) **Atributos de internacionalización, i18n** (forma abreviada en la que 18 es la cantidad de letras entre la i y la n). Se utilizan para aquellas etiquetas cuyo contenido esté en un idioma distinto al indicado para toda la página.

- lang="código" (en documentos HTML y XHTML)
- xml:lang="código" (sólo en documentos XHTML)
- dir="dirección" (donde dirección podrá ser *rtl* o *ltr*)

código se refiere al idioma (es-ES, para español de España; en-GB, para inglés británico; etc.)

dirección hace referencia a la dirección en la que se escriben los caracteres dentro de un párrafo (*right to left* o *left to right*). Se usa en los lenguajes en los que se escribe de derecha a izquierda.

c) **Atributos de eventos**. Permite, a través de un lenguaje de scripts, la ejecución de una determinada acción cuando se produce un evento. Los podemos clasificar en:

- Intrínsicos: onload, onunload.
- De formulario: onblur, onchange, onfocus, onreset, onselect, onsubmit.
- De imagen: onabort.
- De teclado: onkeypress, onkeydown, onkeyup.
- De ratón: onclick, ondblclick, onmouseover, onmousedown, onmousemove, onmouseout.

Ejercicio: Escribir el siguiente ejemplo en el *Bloc de notas* y llamarlo prueba2.html (No vamos a aprender JavaScript, tan solo, para comprender mejor estos atributos, copiad literalmente el código)

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset=UTF-8" />
        <title>prueba2</title>
    </head>
    <body>
        <div onclick="alert('Hola, has pulsado sobre el div')" style="background-color: orange; border: 3px solid
            #C00; position: absolute; top: 200px">
            Pulsa sobre esta capa
        </div>
    </body>
</html>
```

Probar a cambiar el evento onclick, por ondblclick o por onmouseover.

El foco es el elemento de la interfaz gráfica que responde al teclado en un momento concreto. Estos atributos permiten controlarlo y desencadenar acciones cuando un elemento tiene dicho foco. Algunos atributos son:

- onfocus="...", desencadena una acción JavaScript cuando recibe el foco.
- onblur="...", desencadena una acción JavaScript cuando el elemento pierde el foco.

Ejercicio: Escribir el siguiente ejemplo y llamarlo prueba3.html.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset=UTF-8" />
        <title>prueba3</title>
    </head>
    <body>
        <form>
            Haz click sobre el campo de texto <input type="text" onfocus="alert('El campo tiene el foco')"/>
        </form>
    </body>
</html>
```

d) **Atributos de teclado**.

- accesskey="letra", sirve para usar la combinación de teclas *Alt+letra* para acceder de forma rápida a un elemento (atajos de teclado).
- tabindex="n", indica el orden en que se accede a un elemento, utilizando la tecla Tab.

e) **Atributos específicos**. Son los propios de cada etiqueta o grupo de etiquetas.

Aparte de esto, hemos de tener en cuenta que HTML clasifica todos los elementos en dos grupos, atendiendo a la forma en que ocupan los espacios en la página:

a) **inline (en línea**), sólo ocupan el espacio necesario para mostrar sus contenidos. Sólo pueden contener otros elementos *inline* o texto.

b) **block (en bloque)**, realizan un salto antes y otro después del espacio que utilizan para su contenido. Pueden contener otros elementos en bloque, en línea o texto.

Hay algunos elementos que pueden ser en bloque o en línea, según las circunstancias.

4. Descripción detallada de las etiquetas.

4.1. Etiquetas de texto.

Existen muchas etiquetas para representar texto en HTML y aquí indicaremos las más importantes, resumidas en la tabla que aparece a continuación.

Aunque cualquier texto que esté dentro del cuerpo de la página se presentará literalmente, lo usual es usar el elemento párrafo <p> para contener dicho texto. Aunque en HTML no es obligatorio usar el elemento párrafo, en XHTML todo texto debe de estar encerrado en un elemento de párrafo. De esa forma se indica el tipo de texto que es.

Hay una serie de seis etiquetas que se escriben con la letra h seguida de un número del 1 al 6, que sirven para marcar párrafos que se considerarán títulos del texto. De modo que el número 1 marcará títulos de primer nivel, es decir los títulos principales irán marcados con h1. Después se podría usar h2 para designar títulos que se considerarán títulos de segundo nivel y, así, sucesivamente.

Además, existen otros elementos poco utilizados para marcar texto, pero son muy interesantes para dar significado al mismo.

<p>	Atr. básicos, i18n, eventos	No tiene atributos particulares	block
	Es una estructura de bloque que delimita el contenido del párrafo.		
<h1><h2>… <h5><h6>	Atr. básicos, , i18n, eventos	No tiene atributos particulares	block
	Etiquetas de cabecera. Se usan para encabezar las secciones de un documento.		
	Atr. básicos, , i18n, eventos	No tiene atributos particulares	inline
	Enfatiza un bloque de texto. En la mayoría de los navegadores aparece en cursiva.		
	Atr. básicos, , i18n, eventos	No tiene atributos particulares	inline
	Resalta lo más importante de una página. En la mayoría de los navegadores aparece en negrita.		
<blockquote>	Atr. básicos, , i18n, eventos	cite="url"	block
	Indica que el texto es una cita textual normalmente extensa, donde "url" indica de dónde se extrae la cita. El contenido aparece indentado.		
<abbr>	Atr. básicos, , i18n, eventos	title="texto"	inline
	Indica abreviatura, donde "texto" indica el significado de la abreviatura. En la mayoría de los navegadores el contenido aparece subrayado con puntos y presenta un *tooltip* con el "texto".		
<dfn>	Atr. básicos, , i18n, eventos	title="texto"	inline
	Señala definiciones disponibles, donde "texto" indica el significado del término. Presenta un *tooltip* con el "texto".		
 	Atr. básicos	No tiene atributos particulares	Inline /block
	Salto de línea y retorno de carro. En XHTML se escribe 		
<hr>	Atr. básicos, eventos	No tiene atributos particulares	block
	Muestra una línea horizontal. Puede ser usado para separar contenidos. En XHTML se escribe <hr />		
<pre>	Atr. básicos, , i18n, eventos	No tiene atributos particulares	block
	Texto preformateado con fuente no escalable. Respeta los saltos del texto plano.		
<samp>	Atr. básicos, , i18n, eventos	No tiene atributos particulares	block
	Indica un ejemplo de salida por pantalla de un programa informático		
<code>	Atr. básicos, , i18n, eventos	No tiene atributos particulares	inline
	Permite escribir código en algún lenguaje de programación.		

4.2. Codificación en HTML.

En realidad, no todos los sistemas tienen adoptado el sistema de codificación UNICODE. Por eso, en algunas páginas vemos que no se presentan adecuadamente los caracteres que están fuera del ASCII (por ejemplo, la letra ñ).

La mayoría de los navegadores actuales son capaces de decodificar texto en formato ISO-8859-1, además de en UTF-8 y UTF-16, por lo que, en principio, parece que no hay ningún problema para escribir cualquier código.

Sin embargo, cuando alguno de los procesos (el sistema operativo del servidor, el servidor web, el navegador cliente, …) no utiliza la misma codificación (por defecto, UTF-8), se producen conversiones de un sistema de codificación a otro. Muchas de estas conversiones producen errores y, consecuentemente, algunos caracteres no se presentan bien, como ocurre, por ejemplo, con el Bloc de Notas de Windows que cifra, por defecto, usando la tabla no estándar WIN1252.

Por otro lado, en los lenguajes de marcas, no todos los caracteres son utilizables como contenido de las etiquetas. Los caracteres <, >, &, " , ', tienen un significado particular para el lenguaje.

Para solucionar estos problemas de presentación de caracteres nacionales que están fuera del ASCII y de los usados por el propio lenguaje de marcas se usan las llamadas *entidades de caracteres* y *las entidades HTML* respectivamente.

Todas la entidades se escriben con el símbolo & seguido del nombre del código y finalizando con el carácter ;. Por ejemplo ñ es la entidad que representa la ñ. También se puede usar la notación &número; donde número es el correspondiente del carácter en la tabla Unicode. Por ejemplo, para la ñ será ñ

Las *entidades HTML* son:

Carácter	Número	Nombre
"	"	"
'	'	' (el navegador Explorer no la reconoce)
&	&	&
<	<	<
>	>	>

Algunos ejemplos de entidades de carácter son: ñ (ñ), Ñ (Ñ), é (é), É (É), espacio ( ), € (€), ë (ë).

Ejercicio: Escribir el siguiente ejemplo en el Bloc de Notas y llamarlo prueba4.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset=UTF-8" />
        <title>prueba4</title>
    </head>
    <body>
        <p>En este p&aacute;rrafo por fin podemos asegurar que salen las palabras que llevan tildes
            correctamente, cosa que en todos los ejemplos anteriores no</p>
        <p> Adem&aacute;s vamos a escribir matem&aacute;ticamente que 3 es menor que 4:  3&lt;4</p>
```

```
    </body>
</html>
```

4.3. Marcos flotantes.

Los marcos se utilizaban para organizar la presentación de varios documentos HTML simultáneamente en la ventana del navegador. La forma de conseguirlo era dividiendo ésta en diferentes áreas y asignando a cada una de ellas, uno de los documentos a mostrar.

Con la aparición de las hojas de estilos, la utilización de los marcos ha quedado relegada casi por completo. Sólo son válidos en los documentos especificados como tipo *frameset* en su DTD.

En la actualidad se usa otro tipo de marco mediante <iframe> que permite insertar un marco en línea dentro de un documento. El marco insertado se denomina "*flotante*" y puede considerarse como un agujero que se abre en una página web para mostrar otra. <iframe> admite atributos básicos, de internacionalización, de eventos y los siguientes específicos:

- src="url", página que se carga en el marco.
- width="número", anchura en píxeles o en %.
- height="número", altura en píxeles o en %.
- scrolling="**yes**" | "no" | "auto". Indica la aparición o no de barras de desplazamiento laterales.
- frameborder="**1**" | "0". Indica si el marco tiene borde o no.
- name, nombre para el marco.
- marginwidth="número", anchura en píxeles de los márgenes izquierdo y derecho.
- marginheight="número", altura en píxeles de los márgenes superior e inferior.

Ejercicio: Escribir el siguiente ejemplo en el bloc de notas y llamarlo prueba5.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <title>prueba4</title>
    </head>
    <body>
        <h1>Vamos a poner un marco flotante con la página web de un periódico en su interior</h1>
        <iframe src="http://www.ideal.es" width="300" height="300">Su navegador no soporta marcos</iframe>
    </body>
</html>
```

4.4. Etiquetas para enlaces.

Un enlace permite hacer referencia a una URL concreta a la cual se redireccionará al hacer clic sobre él. Se puede definir sobre un texto, una imagen, un elemento de una lista, etc.

Una URL se compone de las siguientes partes:

PROTOCOLO	SERVIDOR	[RUTA]	[ARCHIVO]	[CONSULTA]	[SECCIONES]

http:// https:// ftp:// file://	nombre.subdominio… dominio	Camino de directorios	nombre.extensión	?variable=valor	#ID (ID es el lugar concreto dentro de la página web)

Los caracteres que aparecen en los enlaces deben poder expresarse en ASCII, sin embargo, en ocasiones, las URL contienen caracteres que no se pueden representar mediante este código y, por tanto, deben ser convertidos. La llamada *codificación URL* convierte una URL en un formato ASCII válido.

La codificación URL sustituye estos caracteres especiales por un "%" seguido por los dos dígitos hexadecimales correspondientes al código ISO-8859-1. Como las URL no pueden contener espacios en blanco, cada uno de ellos se sustituye por el signo +. He aquí la codificación de algunos de ellos:

/	?	@	~	ñ	Ñ	Á	Ç
%2F	%3F	%40	%7E	%F1	%D1	%E1	%E7

Las principales etiquetas para crear enlaces son:

a) **<a>**. Etiqueta en línea que permite establecer el origen o el destino del hipervínculo. Dispone de atributos básicos, de internacionalización, de eventos y de foco, Además, cuenta con los siguientes específicos:

- href="url" indica la dirección a la que apunta el enlace. Si la dirección apunta dentro del mismo servidor, se puede emplear una dirección relativa.
- rel="...", indica la relación de la página actual con la página enlazada.
- target="...", señala el marco de destino. Por defecto se toma la ventana actual. Sus posibles valores son: *_blank, _self, _parent, _top.*
- hreflang="...", especifica el código de internacionalización de la página enlazada.

El atributo rel puede tomar los valores: *alternate, start, next, prev, chapter, help*, etc.

El valor de target puede ser el nombre de una ventana o marco, o bien, *_blank*, si se quiere abrir una ventana nueva; *_self (*valor por defecto), si queremos visualizar el documento en el mismo marco ventana; *_parent*, para abrir el enlace en el marco de la página padre de ella y *_top* para abrir la página en el marco superior. Si se pone un nombre que no existe, se abre una ventana nueva.

Ejercicio: Escribir el siguiente ejemplo y llamarlo prueba6.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset=UTF-8" />
        <title>prueba6</title>
    </head>
    <body>
          <a href="#ancla">Pulsa para ir al ancla</a>
          <div style="position: absolute; top: 1000px;">
```

```
            <p id="ancla">Este párrafo está situado muy abajo, y en el hemos situado un ancla,
                mueve la barra de scroll para volver al principio</p>
        </div>
    </body>
</html>
```

b) **<script>**. Esta etiqueta ya la comentamos como etiqueta de cabecera, pero también puede aparecer en el cuerpo.

c) **<link>**. Dicha etiqueta se comentó anteriormente como etiqueta de cabecera.

4.5. Etiquetas para imágenes.

En un sitio web, se pueden usar imágenes de contenido e imágenes de adorno y diseño. Las primeras deben incluirse mediante HTML y las segundas, en la medida de lo posible, con hojas de estilo CSS.

Los formatos de imagen más estandarizados en páginas web son JPG, PNG y GIF.

Existen diferentes etiquetas que permiten insertar imágenes y multimedia:

a) **<img>.** Permite insertar una imagen. Se trata de un elemento en línea que posee atributos básicos, de internacionalización, de eventos y los siguientes específicos:

- src = "url", para indicar la localización de la imagen.
- alt = "texto alternativo", texto que aparece si no se carga la imagen.
- title = "texto descriptivo", título de la imagen.
- width = "px", especifica la anchura de la imagen en píxeles.
- height = "px", especifica la altura de la imagen en píxeles.
- usemap = "#namedelmapa", sirve para asociar la imagen a un mapa de imágenes.

b) **<map>**. Permite crear un mapa de imágenes. Cuenta con atributos básicos, de internacionalización y de eventos y puede ser en línea o de bloque. Se usa junto a la etiqueta <area>.

c) **<area>**. Sirve para especificar una región geométrica de un mapa. Cuenta con atributos básicos, de internacionalización, de eventos y los siguientes atributos específicos:

- shape = "...", para indicar la forma geométrica del trozo (*default*, *rect*, *circle*, *poly*).
- coords = "x...", para señalar las coordenadas del trozo de imagen.
- href = "url", especifica la URL de la dirección enlazada desde el trozo de imagen.
- target ="...", indica dónde se presentará el contenido apuntado en la URL.

Ejemplo:

```
<img src="img.jpg" usemap="#dibujo"/>
<map name="dibujo">
    <area shape="rect" coords="x1, y1, x2, y2"/>
    <area shape="circle" coords="x, y, r"/>
    <area shape="poly" coords="x1, y1, x2, y2, x3, y3, ..."/>
```

```
</map>
```

Ejercicio: Escribir el código que se obtiene tras las siguientes instrucciones y llamarlo prueba7.html.

- Buscar una imagen de un mapa de Andalucía y descargarla al ordenador.
- Entrar en la dirección: http://www.maschek.hu/imagemap/imgmap y subir la imagen de Andalucía
- Generar un mapa web, utilizando las figuras geométricas, para delimitar las provincias, si es necesario utilizar más de una figura para la misma provincia.
- Poner, como destino de cada enlace una página referida a la provincia en cuestión (Por ejemplo: Diputación, Turismo, etc.). Utilizar el atributo alt para poner el nombre de la provincia correspondiente.

d) **<canvas>**. En inglés *canvas* significa lienzo, y es un elemento creado para HTML5 que ha supuesto un mayor dinamismo de las páginas web. Proporciona un área que podremos utilizar para dibujar elementos gráficos mediante lenguaje JavaScript. Eso ha permitido crear juegos, animaciones y elementos visuales atractivos en las páginas web. Sus atributos son:

- id = "...", es el atributo que utilizan todos los elementos HTML para ser identificados. En este caso es casi obligatorio su uso para poder hacer referencia al mismo.
- width = " ", para indicar la anchura del lienzo.
- height = " ", especifica la altura del lienzo.

Una vez que se tiene el elemento, se debe usar código en JavaScript para dibujar en él. Por ejemplo:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8"/>
        <title>Uso Canvas</title>
    </head>
    <body>
        <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
            Your browser does not support the HTML5 canvas tag.
        </canvas>
        <script>
            var c=document.getElementById("myCanvas");
            var ctx=c.getContext("2d");
            ctx.moveTo(0,0);
            ctx.lineTo(200,100);
            ctx.stroke();
        </script>
    </body>
</html>
```

<svg>. SVG es un lenguaje basado en XML que sirve para dibujar gráficos. Está aceptado desde hace tiempo por el W3C, sin embargo, pocos navegadores soportan todavía esta etiqueta, aunque HTML5 sí. La manera de incluir SVG en HTML 5 es mediante la etiqueta <svg>:

```
<svg xmlns="http://www.w3.org/2000/svg">
        etiquetas svg …
</svg>
```

Ejemplo:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" fill="red" />
</svg>
```

4.6. Etiquetas multimedia.

En HTML se establecieron diversos tipos de etiquetas para contenidos multimedia, pero posteriormente, se estandarizaron en una sola, llamada <object>. La mayoría de este tipo de contenidos no los interpreta el navegador directamente, sino que hace uso de pequeños programas llamados *plugins* que se añaden (enchufan) al navegador para tratar con este tipo de elementos complejos.

a) La etiqueta **<object>** tenía la intención de sustituir los elementos <img> y <applet>. Sin embargo, a causa de errores y de la falta de soporte de navegadores, esto no ha sucedido. El soporte para la etiqueta en los navegadores depende del tipo de objeto y, lamentablemente, los principales navegadores utilizan códigos diferentes para cargar el mismo tipo de objeto.

   La etiqueta <object> es una etiqueta contenedora de objetos (video, audio, applets Java, Activex, PDF y Flash), la cual incluye otra, llamada <param/>, que permite pasar los parámetros adecuados al objeto correspondiente. Sus atributos específicos son:

   - data = "url" - Indica la URL de los datos que utiliza el objeto.
   - classid, codebase, codetype - Información específica que depende del tipo de objeto.
   - type - Indica el tipo de contenido de los datos.
   - height = "unidad_de_medida" - Indica la altura con la que se debe mostrar el objeto.
   - width = "unidad_de_medida" - Indica la anchura con la que se debe mostrar el objeto.

   Ejemplo:

```
<object data="El amperio contra Paca.mpeg" type="application/mpeg" />
```

b) **<param>**. A los objetos también se les puede pasar información adicional en forma de parámetros mediante la etiqueta <param> la cual siempre debe estar contenida dentro de <object>. Usa dos atributos:

   - name = "texto" - Indica el nombre del parámetro
   - value = "texto" - Indica el valor del parámetro

   Ejemplo:

```
<object classid="CLSID:XXXXXXXXXX" width="800" height="600" type="video/wmv">
        <param name="FileName" value="archive.wmv" />
        <param name="autoStart" value="true" />
</object>
```

   Uno de los principales inconvenientes de <object> es la forma de incluir vídeos en formato Flash en las páginas HTML. Se puede utilizar el siguiente código:

```
<object data="nombre_video.swf" type="application/x-shockwave-flash"></object>
```

c) **<embed>**. Técnicamente el ejemplo anterior es correcto, pero algunos navegadores como Internet Explorer no visualizan el vídeo en Flash hasta que se ha descargado completamente. Si se trata de un vídeo largo, el usuario tendrá que esperar un tiempo. Para resolver este problema, se usa la etiqueta <embed> dentro de <object>. Esta nueva etiqueta no pertenece a XHTML aunque se ha recuperado para HTML5. Sus atributos son:

- src = "url" - Indica la URL del archivo u objeto que se incluye en la página
- type = "tipo_de_contenido" - Indica el tipo de contenido del objeto (flash, quicktime, java, etc.)
- height = "unidad_de_medida" - Indica la altura con la que se debe mostrar el objeto
- width = "unidad_de_medida" - Indica la anchura con la que se debe mostrar el objeto

Ejemplo:

```
<object width="600" height="400">
    <param name="movie" value="http://www.youtube.com/vafmY67GShgS"></param>
    <param name="wmode" value="transparent"></param>
    <embed src="http://www.youtube.com/ vafmY67GShgS " type="application/x-shockwave-flash"
            width="600" height="400">
    </embed>
</object>
```

El uso de Flash ha dado muy buenos resultados gracias a la potencia de su tecnología pero nos obliga a utilizar una forma de trabajar ajena a XHTML, además de la dependencia de Adobe. En la actualidad la tendencia es eliminar Flash e ir aprovechando las nuevas capacidades de HTML 5 para la multimedia.

En resumen, reproducir audio o video en documentos XHTML es bastante engorroso y complicado debido a los diferentes formatos, codecs, plug-ins, y navegadores. Una buena guía para saber las etiquetas adecuadas en XHTML y sus atributos según diferentes formatos y/o codecs la podemos encontrar en http://www.w3schools.com/html/html_object.asp.

Afortunadamente HTML5 está estandarizando el problema con nuevas etiquetas específicas, aunque el problema de la falta de soporte de los navegadores a diferentes tipos de codecs sigue existiendo.

d) **<video>**. Es una etiqueta nueva que incorpora HTML 5 y es la manera más efectiva para incorporar vídeos a una página web. Usa los siguientes atributos:

- src = "url" - Indica la URL del archivo de video se incluye en la página
- height = "unidad_de_medida" - Indica la altura con la que se debe mostrar el video.
- width = "unidad_de_medida" - Indica la anchura con la que se debe mostrar el video.
- autoplay = "autoplay" – Valor para que el vídeo inicie la reproducción en cuanto se descargue.
- loop = "loop" – Valor para que el vídeo se reproduzca de forma continua.
- controls = "controls" – Valor para que se muestren los controles de reproducción.
- preload = "auto | none | metadata" – Valores para que el vídeo se descargue en cuanto se cargue la página; no se descargue hasta que se pulse *play* o para que sólo se descarguen los metadatos.
- poster = "url" – URL de una imagen que se mostrará cuando el vídeo no se reproduce. Por defecto se usa el primer fotograma del video.

e) **<source>**. Nueva etiqueta HTML5 que permite indicar la URL del vídeo mediante el atributo src y su tipo MIME mediante type. El mecanismo para usar esta etiqueta es codificar el vídeo en varios formatos distintos, luego les hacemos referencia dentro de la etiqueta <video> y el navegador usará el formato que sea capaz de traducir (del que disponga de codecs). Por ejemplo:

```
<video autoplay="autoplay" controls="controls" poster="eufemiano.jpg" >
```

```
    <source src="video1.mp4" type="video/mp4; codecs='avc1.42E01E, mp40a.40.2' " />
    <source src="video1.ogv" type="video/ogg;codecs='theora, vorbis' " />
    Su navegador no es compatible con HTML5
</video>
```

Dentro de type, el uso de codecs es opcional, ya que si el navegador no reconoce el formato no suele hacer caso a los codecs que se indiquen (aunque a veces los descarga). El mecanismo consiste en que, si el primer formato no se reconoce (primer elemento source), se intenta el segundo y, así, sucesivamente. Si ninguno es reproducible por el navegador, éste mostrará la frase final.

f) **<audio>**. Esta etiqueta es análoga a la de <video>, pero dedicada a los archivos de audio. Tiene también los mismos atributos, excepto los referentes al ancho, largo y a la imagen precargada inicial. Con el audio hay el mismo problema con los codecs y formatos, por lo que también se suele convertir el audio a distintos formatos y dar diferentes posibilidades con la etiqueta <source>. Por ejemplo:

```
<audio controls="controls" autoplay="autoplay">
    <source src="audio1.ogg" type="audio/ogg">
    <source src="audio1.mp3" type="audio/mpeg">
    Su navegador no es compatible con HTML5
</audio>
```

4.7. Etiquetas para listas.

Las listas son estructuras que permiten enumerar elementos por lo que se usan en clasificaciones, instrucciones y barras de navegación (menús). Las etiquetas que se utilizan para trabajar con listas son:

	Atr. básicos, de internacionalización, de eventos	block
	Lista sin indicador de orden	
	Atr. básicos, de internacionalización, de eventos	block
	Lista ordenadas (numeradas)	
	Atr. básicos, de internacionalización, de eventos	block
	Elemento de una lista	

He aquí un ejemplo de lista ordenada (numerada) formada por dos elementos:

```
<ol>
    <li>Elemento1</li>
    <li>Elemento2</li>
</ol>
```

Las listas se pueden anidar, es decir, un elemento de una lista puede ser, a su vez, otra lista. Hay que tener cuidado al colocar las etiquetas de cierre y apertura para que no se entrecrucen. Por ejemplo:

```
<ul>
```

```
    <li>Apartado 1
        <ul>
            <li>Sección 1.1</li>
        </ul>
    </li>
    <li>Apartado 2
        <ul>
            <li>Sección 2.1</li>
            <li>Sección 2.2</li>
        </ul>
    </li>
</ul>
```

Existe otro tipo de listas, llamadas listas de definiciones, que permiten indicar elementos y sus definiciones. En ellas se contempla la posibilidad de que un término tenga varias definiciones y que varios términos tengan una definición común. Las etiquetas que utilizan son las siguientes:

<dl>	Atr. básicos, de internacionalización, de eventos	block
	Lista de definición	
<dt>	Atr. básicos, de internacionalización, de eventos	block
	Elemento a definir	
<dd>	Atr. básicos, de internacionalización, de eventos	block
	Definición de un elemento	

Ejemplo:

```
<dl>
    <dt>Término1</dt>
    <dd> Definición1</dd>
    <dt>Término2</dt>
    <dd> Definición2</dd>
    <dd> Definición3</dd>
    <dt>Término3</dt>
    <dt>Término4</dt>
    <dd> Definición3-4</dd>
</dl>
```

4.8. Etiquetas para tablas.

Las tablas permiten representar datos en filas y columnas y, aunque durante cierto tiempo se usaron para maquetar información, esta práctica está totalmente desaconsejada.

Es responsabilidad de quien escribe la tabla asegurarse de que el número de datos se ajusta al de filas y columnas. Si no se hace así, el navegador no informará de los errores, sino que, tan sólo, mostrará huecos en determinados lugares o presentará un comportamiento extraño.

Las etiquetas para tablas son:

a) **<table>**. Inserta una tabla de tamaño variable siendo de tipo bloque. Además de los atributos básicos, de internacionalización y de eventos se pueden usar:

- summary = "texto". Para escribir un breve resumen de los contenidos de la tabla.
- width: anchura en *px* o en porcentaje.
- rules: especifica que líneas de división entre celdas serán visibles. Valores: *none* (valor por defecto), *all*, *rows*, *cols*, *groups*.
- border: especifica el ancho del borde exterior de la tabla en *px*.
- cellspacing: indica el espacio entre celdas en *px*.
- cellpadding: espacio entre el borde una celda y su contenido en *px*.

b) **<tr>**. Crea una fila dentro de la tabla. Tiene atributos básicos, de internacionalización y de eventos.

c) **<td>**. Crea una celda dentro de una fila. Tiene atributos básicos, de internacionalización, de eventos y los principales atributos específicos son:

- abbr = "texto", para escribir un resumen de la celda.
- colspan = "número". Número de columnas que ocupa la celda. (Combinar columnas).
- rowspan = "número". Indica el número de filas que ocupa la celda. (Combinar filas).

Se puede poner una tabla dentro de otra, siempre que se coloque dentro de una celda.

d) **<th>**, posee atributos básicos, de internacionalización, de eventos y los mismos atributos específicos que <td>. Permite indicar las celdas que forman la cabecera de la tabla.

e) **<caption>**, posee atributos básicos, de internacionalización y de eventos. Sólo puede aparecer una vez y se escribe, justo, después de la etiqueta <table>. Se utiliza para poner un título o leyenda a la tabla.

f) **<thead>, <tfoot>, <tbody>**. Son etiquetas para crear tablas de forma avanzada. Su principal utilidad es agrupar filas en la cabecera, el pie o el cuerpo de la tabla y aplicar estilos a cada bloque, sin tener que ir poniendo el estilo en cada fila o celda. Cada grupo de filas debe tener como mínimo una fila definida por <tr>. Las tres poseen los atributos básicos, de internacionalización y de eventos. Además, son de tipo bloque.

Ejercicio: Escribir un documento HTML incluyendo la tabla siguiente. Llamarlo prueba8.html.

```
<table>
    <caption>Tabla de prueba</caption>
    <tr>
        <td rowspan="2" colspan="2"> Hemos unido las dos primeras filas y columnas</td>
```

```
            <td>celda</td>
            <td>celda</td>
        </tr>
        <tr>
            <td>celda</td>
            <td>celda</td>
        </tr>
        <tr>
            <td>celda</td><td>celda</td><td>celda</td><td>celda</td>
        </tr>
        <tr><td>celda</td><td>celda</td><td>celda</td><td>celda</td></tr>
    </table>
```

4.9. Etiquetas para formularios.

Un formulario es un área del documento en la que insertamos elementos de entrada de información, los llamados controles de formulario, que permiten introducir datos, marcar opciones, elegir una opción de un grupo, etc.

Los formularios hacen posible la interacción entre el usuario y un servidor, de forma que el tratamiento de los datos del formulario, lo hace un programa externo al documento que los recibe en el servidor. Estos programas suelen estar escritos en *C, Perl, Java, ASP, PHP*, etc.

a) Para definir un formulario se usa la etiqueta **<form>**, de tipo bloque, que cuenta con atributos básicos, de internacionalización, de eventos y los siguientes específicos:

- action = "url". Indica la página del servidor que va a procesar los datos del formulario (Por ejemplo: páginas PHP, ASP, JSP, Perl, etc.).
- method = "get" | "post". Determina la forma en que el protocolo http envía la información. Con *get*, la información se añade a la URL de la cabecera mediante parejas de la forma URL?name1=value1&name2=value3 mientras que con *post* se envía junto al contenido. Si la información tiene que llegar antes que la página, se envía en la cabecera apareciendo en la barra de direcciones.
- enctype = "application/x-www-form-urlencoded" | "multipart/form-data" | "text/plain".

  Indica el tipo de codificación de los datos antes de enviarlos al servidor. Por defecto, se usa "application/x-www-form-urlencoded" lo que significa que todos los datos usarán la codificación URL (los espacios se sustituyen por el símbolo "+", y los caracteres especiales se convierten en su valor ASCII expresado en hexadecimal). Con el valor "multipart/form-data" no se codifican los caracteres, tan solo se usa cuando se envían archivos adjuntos al formulario. Con "text/plain" sólo los espacios en blanco se sustituyen por el signo "+".

  La información importante no debe enviarse con el método *get*, a menos que vaya codificada. Por tanto, si los datos a enviar son de control, se usa *get*. Pero si los datos son sensibles o se van a almacenar en una base de datos o en un fichero, usaremos *post*.

Dentro de la etiqueta <form>, se colocan los controles, cuyas etiquetas se nombran a continuación.

b) **<input>**. Etiqueta en línea con atributos básicos, de internacionalización, de eventos, de foco y los siguientes:

- type = "...", con posibles valores siguientes:

- *text*, define un campo o cuadro de de texto para editar.
- *password*, define un campo de texto para enmascarar (se usan * para sustituir los caracteres)
- *checkbox*, define una casilla de verificación.
- *radio*, define un botón de radio.
- *submit, d*efine un botón de acción para enviar los datos del formulario.
- *reset*, define un botón que "resetea" todos los valores a los valores por defecto.
- *file, d*efine un campo de texto para teclear el nombre de un archivo y un botón para explorar el archivo que se enviará al servidor.
- *hidden, d*efine un campo de texto donde los caracteres quedan ocultos o invisibles.
- *image*, define una imagen como botón de acción para enviar los datos del formulario.
- *button*, define un botón para programarlo mediante eventos.

Dependiendo del tipo anterior, se pueden utilizar otros atributos, como son:

- value = "texto", indica el valor del control. Para cuadros de texto es el valor inicial que aparece en el recuadro. Para los botones de tipo *radio* y los *checkbox*, es el valor que se envía al servidor. Para botones de acción, *button*, es el título del botón.
- size = "número", En los controles *text* y *password* especifica el ancho del control en número de caracteres. (Por defecto son 20).
- maxlength = "número", En los controles *text* y *password* indica la longitud máxima de caracteres que se puede teclear.
- checked = "checked" indica si el control está marcado en los controles *radio* y *checkbox*.
- disabled = "disabled" desactiva el control y, por tanto, no se envía su valor al servidor.
- readonly = "readonly" pone el control en sólo lectura. (Sólo para los controles *text* o *password*)
- src = "url", para indicar la ruta de la imagen que actuará como un botón de envío.
- alt = "texto", descripción o texto alternativo que aparece cuando no se carga la imagen cuando el control es de tipo *image*

c) **<fieldset>** Sirve para hacer agrupaciones lógicas de controles dibujando un marco que engloba varios de ellos. Es de tipo bloque y tiene atributos básicos, de internacionalización y de eventos.

d) **<legend>** Se escribe justo debajo de la apertura del <fieldset> para poner un título o rótulo al grupo.

e) **<label>**, tiene atributos básicos, de internacionalización y de eventos. Define una etiqueta que puede asociarse con un control de formulario mediante el atributo específico for="id", siendo id el identificador del control del formulario al que se asocia. Mejora la entrada de usuario, ya que si se hace un clic de ratón en el texto que muestra <label> conmuta el estado del control asociado.

f) **<textarea>** Define un cuadro de texto multilínea. Es de tipo en línea, con atributos básicos, de internacionalización, de eventos, de foco y los siguientes específicos:

- cols = "número" Número de columnas.
- rows = "número" Número de filas.

- disabled = "disabled" Desactiva el <textarea> con lo que no se envía información.
- readonly = "readonly", de sólo lectura por lo que no se puede escribir.

g) **<select>** Muestra listas desplegables o desplegadas. Es de tipo en línea y admite atributos básicos, de internacionalización, de eventos, de foco y los siguientes específicos:

- size = "número" fija el tamaño inicial de la lista. Si es mayor que uno la lista aparece desplegada.
- multiple = "multiple" permite elegir más de una opción de la lista.
- disabled = "disabled", desactiva el control.

h) **<option>** Indica cada una de las opciones de una lista desplegable. Admite los siguientes atributos específicos:

- selected = "selected", opción marcada por defecto.
- value = "value", valor de la variable de la opción correspondiente que se envía al servidor.
- disabled = "disabled", desactiva esa opción de la lista.

i) **<optgroup>**, sirve para englobar etiquetas <option> y hacer subgrupos. Tiene atributos básicos, de internacionalización, de eventos y los siguientes específicos:

- label = "texto", texto para la cabecera del grupo.
- disabled = "disabled", desactiva el grupo.

Ejemplo de uso de <optgroup>:

```
<select>
  <optgroup label="vehículos suecos">
        <option value="Volvo">Volvo</option>
        <option value="Saab">Saab</option>
  </optgroup>
  <optgroup label="vehículos alemanes">
        <option value="Opel">Opel</option>
        <option value="Audi">Audi</option>
        <option value="VW">Volks Wagen</option>
  </optgroup>
</select>
```

4.9.1. Nuevos controles de formulario en HTML5.

HTML5 mejora los formularios gracias a la funcionalidad de sus nuevos controles. No todos los controles son soportados por los navegadores, pero cuando un navegador no lo soporta, se suele presentar como un cuadro de texto normal.

Algunos de ellos son los cuadros de texto especializados que permiten editar datos de diferentes tipos y validarlos. Para ello se tienen nuevos valores para el atributo type de la etiqueta <input>.

- input type="number". Acepta sólo números. El atributo maxlength permite indicar un valor máximo para el cuadro.
- input type="email". Acepta sólo direcciones de correo electrónico.
- input type="url". Acepta sólo direcciones URL.
- input type="date". Acepta sólo fechas válidas. Usa el formato de fecha configurado en el sistema operativo del usuario que visita la página. Los navegadores además proporcionan un cuadro visual más sencillo para recoger la fecha.
- input type="time". Acepta sólo horas válidas; funciona igual que el cuadro anterior.
- input type="datetime". Acepta fecha y hora.
- input type="month". Acepta sólo números del 1 al 12, referidos a un mes.
- input type="search". Presenta un cuadro de texto pensado para hacer búsquedas.
- input type="tel". Permite introducir números de teléfono.
- input type="range". Presenta un control para elegir datos entre un rango. Los atributos max y min establecen el rango máximo y mínimo del control. El atributo step indica el valor del incremento.
- input type="color". Presenta un control de selección de colores. El color se toma en formato #xxxxxx donde cada x es una cifra hexadecimal.

De igual forma aparecen nuevos atributos:

- required = "required". Este atributo obliga a rellenar con algún valor el control en el que se usa. Es decir, hace que un determinado control sea de obligado rellenado en un formulario.
- pattern ="expresión regular" Permite colocar una expresión regular en un cuadro de texto que, obligatoriamente, tendrá que cumplir el cuadro en el que se use el atributo. Ejemplo (cuadro de texto que sólo acepta introducir 5 letras mayúsculas y tres números):
- placeholder = "texto" Es un texto que ayuda a rellenar un cuadro de un formulario (está especialmente pensado para los cuadros de texto) colocando un texto inicial en el cuadro que se va en cuanto el cuadro obtiene el foco del usuario. Ejemplo:

```
<input type="text" pattern="[A-Z]{5}[0-9]{3}" id="d1" name="d1" placeholder="5 letras y tres números" />
```

- autocomplete = "on | off" Permite activar o desactivar el autocompletado del navegador. El autocompletado es la opción que permite a los usuarios cuando rellenan un formulario ver entradas habituales que han escrito en el mismo u otros formularios.
- min, max y range. Atributos que se pueden utilizar en muchos tipos de cuadros (number, date, time, range,..) que establecen los límites del cuadro y el mínimo incremento de valor en el cuadro.

5. Validación.

La validación es un procedimiento que consiste en comparar un documento con un modelo que especifica las normas para escribir en un determinado lenguaje. Dicho modelo es lo que denominamos al inicio del tema una DTD, una Definición de Tipo de Documento.

La validación no es estrictamente necesaria, de hecho, hay cientos de miles de páginas de Internet que no son válidas, pero es bastante recomendable validar los documentos, sobre todo al comienzo.

La validación podemos realizarla de dos maneras: en línea como el validador del W3C o usando herramientas instalables de forma local. Estas herramientas pueden ser mediante complementos en el navegador, entornos de desarrollo o con validadores específicos.

5.1. Validación en línea.

El consorcio W3C dispone de una herramienta gratuita en Internet en http://validator.w3.org/. La herramienta proporciona tres formas de operar.

- Validate by URI. Valida la página cuya dirección se suministre.
- Validate by Upload. Se presenta un formulario mediante el cual se puede subir el archivo con el contenido de la página a validar.
- Validate by Direct Input. Valida el contenido insertado directamente en un cuadro de texto.

Si la página no pasa la prueba de validación, se muestra un listado con los errores y una ayuda para resolverlos. En el caso de que sea válida, podemos insertar una imagen que informa a los usuarios de que la página es válida. Para ello, sólo hay que añadir el código que nos proporciona:

```
<a href = "http://validator.w3.org/check?uri=referer">
    <img src = "http://www.w3.org/Icons/valid-html401"
    alt = "Valid HTML 4.01 Transitional" height = "31" width = "88"/>
</a>
```

5.2. Herramientas de validación.

En el navegador *Firefox* podemos instalar desde https://addons.mozilla.org/es/firefox/addon/html-validator/ un complemento para validar llamado HTML Validator. (La disponibilidad del complemento dependerá de la versión de *Firefox* instalada)

Tras la instalación y el reinicio del navegador se muestra la ventana de configuración. En ella se pide al usuario el tipo de validación que quiere realizar. Las opciones son:

- *HTML Tidy*. Mejor para HTML y ofrece ayuda para resolver errores.
- *SGML Parser* es el mismo validador que el del W3C y ofrece menos ayuda que al anterior.
- *Serial*. Realiza las dos validaciones de forma consecutiva.

Configurado el validador, cuando abramos cualquier página web, en la esquina superior derecha del navegador se mostrará un icono indicando si la página es válida o no. Si no lo es, aparecerá el icono en forma de cruz y colocando el puntero del ratón se presenta un mensaje con el número de errores y advertencias. Haciendo doble clic se muestran todos los errores de forma detallada.

Si no disponemos de conexión a Internet, podemos instalar un validador sencillo en nuestro equipo como es TidyGUI. Una vez instalado, sólo hay que introducir el archivo fuente y ejecutar Tidy!.

Algunas herramientas de diseño incorporan validadores, como es el caso de la aplicación de diseño Dreamweaver de Adobe Systems Incorporated.