Tema 2. HTML, XHTML y HTML5

Tema 2. HTML, XHTML y HTML5.

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, XHTML 1.0 y HTML 5.

Pasar de HTML 4.01 a XHTML 1.0 es bastante fácil: solo tendremos que asegurarnos de cumplir unos pequeños detalles para lograrlo. Aunque en la actualidad casi todo lo nuevo que se desarrolla se escribe en HTML 5, hay que recordar que existen innumerables aplicaciones web escritas tanto en HTML 4.01 como en XHTML 1.0, que hay que actualizar y mantener, por lo que es necesario conocerlas.

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, utilizaremos el lenguaje CSS (*Cascade Style Sheet*) que estudiaremos en el siguiente tema.

Poco después de sus inicios, HTML empezó a crecer de una forma descontrolada con etiquetas nuevas y funcionalidades propias de cada navegador. Debido a esta falta de estandarización se propuso utilizar como base XML para imponer ciertas reglas, pero usando las etiquetas de HTML. Así nació XHTML.

La diferencia principal entre XHTML y HTML es que XHTML al ser un derivado de XML genera documentos **bien formados**, los cuales deben cumplir para ello las siguientes reglas:

a) Cada elemento debe contar con su marca de apertura y de cierre, así como la anidación de elementos debe ser correcta.
b) Debe existir un único elemento raíz, que en XHTML se debe ser <html>.
c) 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).
d) Las etiquetas siempre se escribirán en minúscula.
e) Los atributos se escriben con minúscula, seguidos de un signo igual y de un valor delimitado por comillas simples o dobles. Por ejemplo, style="valor".
f) Los elementos vacíos deben tener tanto la etiqueta de apertura como la de cierre, o bien usarse la notación abreviada. (Así en HTML la etiqueta
 es correcta, pero en XHTML deben escribirse como
</br> o usando la notación abreviada
.)

Con estas sencillas normas nos aseguramos que nuestros documentos HTML 4.01 serán también documentos XHTML válidos. Es muy importante comprender que las normas que definen cómo se deben interpretar las etiquetas HTML así como las reglas de estilos, 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, algunos navegadores suelen 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 su correcta visualización por la mayoría de los usuarios.

1.1. Declaración <!DOCTYPE>

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

La declaración <!DOCTYPE> indica: el tipo de documento, la versión de HTML o XHTML que se usa y una URL indicando la DTD (Definición de Tipo de Documento) para dicho 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, las que son opcionales, si se pueden repetir, cuántas veces, etc. y cómo deben usarse. Una manera de definir dicha gramática es mediante un DTD.

Así pues en un DTD se indica la estructura de un documento mediante reglas, de esta manera los navegadores al conocer el 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>. Los DTD disponibles en las Recomendaciones del W3C para los documentos HTML y XHTML estandarizados más actuales 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. Hasta hace poco era 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

Este 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">
```

1.1.5. XHTML 1.0 Transitional

Esta DTD contiene todos los elementos HTML y atributos, incluyendo elementos de presentación y obsoletos (como las fuente). Los marcos no están permitidos. Además el documento se debe escribir como XML **bien formado**. 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

Este DTD es igual a XHTML 1.0 Transitional, pero permite la utilización de marcos. El documento se debe escribir como XML **bien formado.**

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

1.1.7. XHTML 1.1

Este 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).

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

Además, en todos los documentos de tipo XHTML, en la etiqueta raíz <html> se debe especificar el espacio de nombres (atributo xmlns) y el idioma en que está escrita la página (atributo xml:lang). En el siguiente ejemplo hemos usado como idioma el español.

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

1.1.8. HTML5

La especificación HTML5 no está basada en SGML y por ello no existe una referencia al DTD que describa el tipo de documento; sin embargo se requiere la siguiente declaración:

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

La etiqueta raíz <html> también es más sencilla al no tener que indicar un espacio de nombres.

Nota 1. En los DTD estrictos, el texto no debe ser insertado directamente en el cuerpo (dentro de la etiqueta <body>); ni se deben insertar elementos de bloque dentro de elementos de línea.

Nota 2. En http://www.webstandards.org/learn/reference/templates/ podemos encontrar plantillas para todos los diferentes tipos de documentos vistos, excepto para HTML5.

2. Estructura de una página web.

La estructura básica de una página web cambia según sea HTML 4.01, XHTML o HTML5. Como son tantas las posibilidades, hablaremos sólo de las dos versiones más populares actualmente: XHTML 1.0 transicional y HTML5. Ambas representan muy bien tanto a XHTML como a HTML respectivamente, lo que nos permite inducir el esqueleto de una página web en cualquier otra versión.

La estructura de una página XHTML 1.0 transicional que esté escrita en español es la siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>Título de la página</title>
    </head>
    <body>
        Contenido de la página web
    </body>
</html>
```

- La etiqueta <!DOCTYPE> indica el tipo de documento, la versión de HTML/XHTML que se usa y el DTD (Definición de Tipo de Documento). De esta forma, los clientes web conocerán sin ambigüedad con qué reglas deben procesar el documento. En caso de elegir otra versión de XHTML simplemente hay que cambiar el tipo y las rutas al documento validador (el DTD).
- Elemento <html> marca la raíz de la página indicando que todo el texto contenido entre ambas es código HTML. Por tanto, <html> debe abrir este tipo de documento y </html>, cerrarlo. Además se debe especificar el espacio de nombres al que pertenecen las etiquetas (atributo xmlns) y el lenguaje en el que está escrito el documento (atributo xml:lang).

  Es conveniente utilizar también 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 colocan el título de la página (elemento <title>), los elementos que sirven para incluir estilos CSS, código JavaScript y otros elementos de información o complementarios que se utilizarán dentro de la página.
- La etiqueta <meta> indica el tipo del documento (texto en formato html), y el sistema de codificación del texto. En el código anterior se utiliza el más recomendable: el UTF-8.
- Elemento <body> contiene el cuerpo de la página, es decir todo el contenido que hará visible el agente de usuario (navegador, explorador, etc.)

En el caso de HTML5 la estructura es muy parecida pero simplifica muchas etiquetas:

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

Las diferencias más importantes respecto a XHTML son:

- La etiqueta <!DOCTYPE> está simplificada.
- La etiqueta <meta> es mucho más sencilla para indicar el sistema de codificación.
- El lenguaje a usar en la página se indica sólo con el atributo lang del elemento <html>.

2.1. Cabecera.

En la cabecera de un documento HTML/XHTML (<head>) 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 indica, 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 complementaría 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 qué tipo de recurso, href para indicar cuál es el recurso, y 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 el 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 (*favicon*). Se suele usar icon para iconos de tamaño 60x60, mientras que se usa shortcut 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 que aparecerá en la barra de título de la ventana. Es importante no confundirse con el texto de encabezado que suelen presentar las páginas web. Estos títulos o encabezados se escriben en el <body> con una etiqueta de encabezado del tipo <hn> delante 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. Los atributos son:

- name. Permite indicar 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, generador, organization, rating, robots, etc.
- content. Contiene el valor de la propiedad indicada por el atributo name.

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

```
<meta name="author" content="Paquito" />
<meta name="keywords" content="instituto, zaidín, clase, ciclo, pardillos" lang="es" />
<meta name="description" content="Ejemplos para la explicación en clase" lang="es"/>
<meta name="robots" content="index, nofollow" lang="es"/>
```

- http-equiv. Indica una propiedad al navegador en forma de cabecera HTTP como si el propio servidor web la hubiera generado. Se utiliza conjuntamente con el atributo content para proporcionar parejas de valores. Posibles valores: 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" /> (No en HTML5)
<meta http-equiv="content-language" content="es_Es" />
<meta http-equiv="refresh" content="10" />
```

- charset. Sólo en HTML5, y sirve para indicar la codificación del texto. Su papel es sustituir la versión larga http-equiv="content-type" que se usa en HTML 4.01 y XHTML. Ejemplo:

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

En http://www.metatags.info/ se puede encontrar información adicional sobre la etiqueta <meta/>.

e) **<script>** Con esta etiqueta podemos incluir código en algún lenguaje cliente (código que ejecuta el navegador), como por ejemplo, JavaScript. Esta etiqueta también puede aparecer una o más veces en la sección <body> aunque no suele ser frecuente. Se pueden usar los siguientes atributos:

- type. Indica el tipo MIME del contenido que se colocará dentro de la etiqueta. Si se usa JavaScript el valor es text/javascript y es el que se asume por defecto si no se indica el atributo.
- charset. Codificación usada para el texto del script. Por defecto toma el definido para la página.
- src. Permite indicar la URL a un archivo externo que contendrá el código del script que se colocará o insertará en la página.

Por ejemplo:

```
<script type="text/javascript"> alert("hola") </script>
<script type="text/javascript" src="/js/banner.js"></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>
```

Como posteriormente comprobaremos, también es posible asignar estilos directamente 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 dos 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 dicho contenido se le pueda dar un formato especial mediante CSS.

b) permite agrupar varios elementos dentro de un bloque y de esta forma 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 asignar un identificador único a la zona o capa para luego referenciarla.

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 el uso de estas etiquetas veremos 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 de que la posición del elemento sea absoluta, los desplazamientos se miden desde los bordes del documento, Si es relativa, desde la posición en que el navegador colocaría dicho elemento.

Por ejemplo:

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

Además, se pueden aplicar otras muchas 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 y llamarlo prueba1.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" >
        <title>Prueba 1</title>
    </head>
    <body>
        <h1>Vamos a poner una etiqueta span y una div</h1>
        <span style="color:red;">Esto se incluye en una etiqueta span. No provoca salto de linea</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>
```

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, a las fuentes o a los tipos de letra.

El formato general de una etiqueta es:

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

Los atributos de se pueden clasificar en:

a) **Atributos básicos**. Se pueden usar en casi todas las etiquetas. Su mayor utilidad se consigue cuando se emplean trabajando conjuntamente con hojas de estilo y/o scripts. Son los siguientes:
   - id. Identifica de forma única cada elemento de un documento HTML. Su valor debe empezar por letra y no tener espacios aunque sí puede tener números. El valor asignado no se puede repetir en ninguna etiqueta del mismo documento.
   - class. Determina clases de etiquetas para las hojas de estilo. (Lo veremos…)
   - style. Establece, de forma directa propiedades de estilo CSS a un elemento concreto.
   - title. Establece el título de un elemento que se suele mostrar en un pequeño recuadro emergente cuando se pasa el ratón por encima de dicho elemento.

b) **Atributos de internacionalización (i18n).** Se encuentra así escrito en muchos lugares: el 18 es por la cantidad de letras entre la i y la n, y sirve para abreviar). Se utilizan para aquellas etiquetas cuyo contenido se expresa en un idioma distinto al indicado en la cabecera 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 de texto. Su valor por defecto es *ltr* (acrónimo de *left to right*, de izquierda a derecha).

c) **Atributos de eventos**. Con estos atributos se permite a través de un lenguaje de script, la ejecución de una determinada acción cuando se produce un suceso o 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 llamándolo prueba2.html (No vamos a aprender JavaScript, tan solo lo usaremos para comprender mejor estos atributos.)

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" >
        <title>Prueba 2</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>
```

Como ejercicio se puede probar a cambiar el evento onclick, por ondblclick o por onmouseover.

En el diseño de interfaces gráficas, el foco es el elemento de la interfaz gráfica que responde al teclado en un momento concreto. Los siguientes atributos permiten controlarlo y desencadenar acciones cuando un elemento obtiene o pierde dicho foco.

- 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>Prueba 3</title>
    </head>
    <body>
          <form>
              Haz click sobre el campo de texto <input type="text"  onblur="alert('¿Has terminado?')"/>
          </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. (Los llamados 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. A medida que vayamos estudiando las diferentes etiquetas, iremos haciendo lo mismo con sus atributos.

Es importante, antes de empezar a estudiar las diferentes etiquetas, tener en cuenta que HTML las clasifica en dos grupos atendiendo a la forma en que ocupan los espacios en una página:

- **inline (en línea**), sólo ocupan el espacio necesario para mostrar sus contenidos. Sólo pueden contener otros elementos *inline* o texto.
- **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. Otras están resumidas en la tabla que aparece a continuación.

Aunque cualquier texto que esté dentro del cuerpo de la página se presentará literalmente, lo correcto 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 comienzan con la letra h, seguidas 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.

Existen elementos que permiten marcar el texto de forma especial. Así permite generar el efecto de negrita; el efecto de cursiva; <sub> para presentar subíndices y <sup> superíndices.

Existen otros elementos menos utilizados para marcar texto, pero son muy interesantes para dar significado al mismo. Muchos de ellos no tienen efectos visuales cuando se utilizan, pero se espera que en el futuro haya cada vez más herramientas software que sean capaces de manipular de forma adecuada estos elementos y así poder dar más posibilidades al texto escrito de las páginas web.

<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".		
<acronym>	Atr. básicos, , i18n, eventos	title="texto"	inline
	Indica acrónimos, donde "texto" indica el significado completo de la siglas. En la mayoría de 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 indicar que el contenido es código fuente 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 (mal escritas), podemos observar cómo caracteres que están fuera del ASCII, como puede ser la letra ñ, no se presentan adecuadamente.

Los navegadores actuales son capaces de poder 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 el código que sea.

Pero si algunos de los procesos (el sistema operativo del servidor, el propio servidor web, el navegador cliente,…) no utilizan 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 adecuadamente. Por ejemplo, Windows en su Bloc de Notas usa por defecto la tabla de codificación WIN1252 que no es estándar.

Por otro lado, en los lenguajes de marcas, no todos los caracteres pueden usarse como contenido de las etiquetas. Los caracteres <, >, &," , ', tienen un significado para el lenguaje y no se pueden emplear.

Para solucionar los problemas de presentación de caracteres nacionales que están fuera del ASCII y de la prohibición de emplear aquellos caracteres que son usados por el propio lenguaje de marcas, se utilizan las llamadas *entidades de caracteres* y las *entidades HTML* respectivamente.

Todas la entidades de ambos tipos se forman con el símbolo & seguido del nombre del código y finalizando con ;. Por ejemplo ñ es la entidad de carácter 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 &241;

De igual forma 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). En http://ascii.cl/es/codigos-html.htm podemos encontrar una tabla con todas ellas.

Ejercicio: Escribir el siguiente ejemplo y llamarlo prueba4.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" >
        <title>Prueba 4</title>
    </head>
    <body>
        <p>En este p&aacute;rrafo por fin podemos asegurar que las palabras que llevan tildes aparecen
            siempre 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. Etiquetas para marcos.

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

Con la aparición de las hojas de estilos, la utilización de los marcos ha quedado relegada casi por completo. Importante: los marcos sólo son válidos en los documentos especificados como tipo *frameset* en su DTD y no es soportada en HTML5.

Un documento con marcos no tiene sección body. En su lugar se usa la sección frameset, y una sección noframes para los casos en que el navegador no pueda visualizar los marcos.

Su declaración tiene esta estructura:

```
<frameset [cols="..."| rows="..."]>
    <frame src="url1" name="nombre_marco1"/>
    <frame src="url2" name="nombre_marco2"/>
    ...
    <noframes> Texto que aparece si el navegador no acepta marcos </noframes>
</frameset>
```

El elemento <frameset> contiene uno o más elementos <frame>, en donde cada <frame> puede contener un documento distinto. En la etiqueta <frameset> mediante atributos se debe especificar cuantos marcos contiene y cómo se disponen, si en filas o columnas.

Con rows especificamos los marcos horizontales mediante una lista de valores separados por comas. Los valores son longitudes expresados en *px* o porcentajes. Por ejemplo rows="20%,*,40%".

Con cols se especifican los marcos verticales de igual forma.

La etiqueta <frame> aparecerá tantas veces como filas o columnas se hayan indicado y, en ella, se especificará toda la información relativa a un marco. El atributo src se usa para indicar la URL de la página que se va a cargar en él y el atributo name para nombrar o identificar el marco.

Opcionalmente se pueden usar los atributos frameborder (0/1), marginwidth, marginheight, noresize y scrolling (auto/yes/no).

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", nombre para identificar el marco.
- marginwidth="número", anchura en píxeles de los márgenes superior e inferior.
- marginheight="número", altura en píxeles de los márgenes izquierdo y derecho.

Ejercicio: Escribir el siguiente ejemplo y llamarlo prueba5.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" >
        <title>Prueba 5</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">Página no disponible</iframe>
    </body>
</html>
```

4.4. Etiquetas para enlaces.

Un enlace permite hacer referencia a una URL concreta a la cual se redireccionará el navegador 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 tienen que 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.

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

La codificación URL sustituye estos caracteres especiales por un "%" seguido por los dos dígitos hexadecimales correspondientes al código ISO-8859-1. Además, como las URL no pueden contener espacios en blanco, éste se sustituyen por el signo + o por su equivalente %20. Los propios navegadores son los encargados de realizar la codificación URL cuando sea necesario.

Las principales etiquetas para crear enlaces son:

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

- name="valor". (En HTML5 no es válido y debe usarse en su lugar el atributo id="valor"). Sirve para nombrar una marca dentro de un documento HTML que será el destino de un enlace.
- 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 el tipo de relación de la página actual con la página enlazada.
- rev="...", indica el tipo de relación de la página enlazada con la actual. (Sin soporte en HTML5)
- target="nombre_marco", señala el marco de destino. Si no se usa, se toma por defecto la ventana actual. Otros posibles valores son: *_blank, _self, _parent, _top.*
- hreflang="...", especifica el código de internacionalización de la página enlazada.

Los atributos rel y rev pueden 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 o 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>Prueba 6</title>
    </head>
    <body>
          <a href="#destino">Pulsa aquí para ir al párrafo de abajo</a>
          <div style="position: absolute; top:1000px">
              <p><a id="destino">Este</a> párrafo está situado muy abajo, y en el hemos situado el destino de
                   enlace.  Mueve la barra de scroll para volver al principio del documento, o si quieres
                   <a href="http://www.w3schools.com">pulsa aquí para visitar W3Schools.com</a>
              </p>
          </div>
    </body>
</html>
```

b) **<script>**. Esta etiqueta ya la comentamos como etiqueta de cabecera para incluir a través de un archivo código de un script, 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 otros objetos 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 descriptivo", texto que aparece si no se carga la imagen. (!Usar siempre...!).
- 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 = "#namedelmap", sirve para asociar la imagen a un mapa de imágenes.

Es importante tener en cuenta que las imágenes tienen un comportamiento en línea.

b) **<map>**. Permite crear un mapa de imágenes de forma que el mapa puede dividirse en áreas o zonas que se pueden *clickear*. 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 de la zona (*default, rect, circle, poly*).
- coords = "x...", para señalar las coordenadas de la zona de la imagen.
- href = "url", especifica la URL de la dirección enlazada desde el área o zona de la imagen.
- target = "...", indica dónde se presentará el contenido apuntado por el enlace.

Ejemplo:

```
<img src="img.jpg" alt="mapa" usemap="#dibujo"/>
<map name="dibujo" id="dibujo">
    <area shape="rect" coords="x1, y1, x2, y2" href="http://www.sitio.es" target="_blank"/>
    <area shape="circle" coords="x, y, r" href="carpeta/archivo1.html" target="_self"/>
    <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.
- Entrad en la dirección: http://www.maschek.hu/imagemap/imgmap y subid 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.
- Sin poner página de destino, tan solo 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 identificarles. 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 para dibujar, se usa código JavaScript para dibujar en él. Ejemplo:

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

e) **<svg>.** SVG es un lenguaje basado en XML que sirve para dibujar gráficos. Está aceptado desde hace tiempo por la W3C, sin embargo no todos los 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 existen para los contenidos multimedia diversos tipos de etiquetas que 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* o complementos que se encargan de tratar con este tipo de elementos complejos.

a) <object>. En (X)HTML la etiqueta <object> se usa como contenedora de muchos tipos de objetos (video, audio, applets Java, ActiveX, PDF y Flash), pero en HTML5 el audio y el video tienen su propia etiqueta El soporte para esta etiqueta por parte de los navegadores depende del tipo de objeto a contener, y lamentablemente, los principales navegadores utilizan códigos diferentes para cargar el mismo tipo de objeto.

Los atributos específicos para <object> son:

- data = "url" - Indica la URL de los datos que utiliza el objeto.
- name="nombre" - Indica un nombre para el objeto.
- type ="tipo de objeto" - 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.
- classid, codebase, codetype - Información específica que depende del tipo de objeto concreto. Estos tres últimos atributos no son válidos en HTML5.

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 parámetros:

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

Los posibles valores de estos dos parámetros va a depender del tipo del objeto en cuestión. Ejemplo:

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

c) **<embed>**. Esta etiqueta también se puede usar como contenedor para objetos externos o complementos (*plugins*). Esta etiqueta no pertenece a (X)HTML sino sólo a HTML5, aunque muchos navegadores la implementaban en (X)HTML "a su manera". 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:

```
<embed src="http://www.youtube.com/v/6oZ8SfsNRhI"
        type="application/x-shockwave-flash"
        width="600"  height="400" />
```

Para el audio y video, 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 HTML, además de la dependencia de Adobe Systems. En la actualidad la tendencia es ir eliminando Flash e ir aprovechando las nuevas capacidades de HTML5 para la multimedia, aunque el problema de la falta de soporte de los navegadores a diferentes tipos de códec sigue existiendo.

En resumen, la incrustación de objetos en HTML es bastante engorrosa y complicada debido a los diferentes formatos, codecs, plug-ins, y navegadores. Pero si trabajamos en HTML5, lo más conveniente es usar la etiquetas <object> y <embed> exclusivamente para aquellos objetos que no sean audio ni video, y usar las nuevas etiquetas específicas <video> y <audio> de HTML5 para estos tipos de contenidos.

(Una buena guía para saber las etiquetas adecuadas en HTML y sus atributos según diferentes formatos y/o codecs la podemos encontrar en http://www.w3schools.com/html/html_object.asp.)

d) **<video>**. Es una etiqueta nueva que incorpora HTML5 y es actualmente la manera más efectiva para incorporar vídeos a una página web. Por ahora, los únicos códec que soporta son MP4, WebN y Ogg. Usa los siguientes atributos:

- src = "url" - Indica la URL del archivo de video a incluir 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 solo se descargue los metadatos.
- posters = "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 video mediante el atributo src y su tipo MIME mediante type. El mecanismo típico de uso de esta etiqueta es codificar el vídeo en varios formatos distintos, luego con <source> hacemos referencia a estos vídeos dentro de la etiqueta <video> y el navegador usará el formato que sea capaz de traducir (aquel del que disponga su codec). 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 códec es opcional, ya que si el navegador no reconoce el formato no suele hacer caso a los códec que se indiquen (aunque a veces los descarga). El mecanismo es 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. Por ahora, los únicos códec que soporta son MP3, Wav y Ogg. Con el audio hay el mismo problema con los códec 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 Lista sin indicador de orden	block
	Atr. básicos, de internacionalización, de eventos Lista ordenadas (numeradas)	block
	Atr. básicos, de internacionalización, de eventos Elemento de una lista	block

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

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, es decir de forma tabular 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 de la tabla expresada en *px* o en porcentaje.
   - rules: especifica qué líneas de división entre celdas serán visibles. Valores: *none* (valor por defecto), *all*, *rows*, *cols*, *groups*. HTML5 no lo soporta, en su lugar se usa CSS.
   - 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. (Combina columnas).
   - rowspan = "número". Indica el número de filas que ocupa la celda. (Combina 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. Mediante el atributo align (valores *left*, *right*, *top*, *bottom*) se indica el lugar respecto a la tabla donde aparece el contenido.

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 border="1">
    <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><td>celda</td>
    </tr>
    <tr><td>celda</td><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. y posteriormente enviar dichos datos a un servidor web para que use dichos datos.

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. (Normalmente páginas escritas en un lenguaje de servidor: PHP, ASP, JSP, Perl, etc.).
- method = "get | post". Determina la forma en que el protocolo HTTP envía los datos al servidor. Con *get*, la información se añade a la URL de la cabecera mediante parejas de la forma URL?name1=value1&name2=value2 mientras que con *post* se envía junto al contenido de la respuesta.

  La información importante no debe enviarse con el método *get*, a menos que vaya encriptada. Por tanto si los datos a enviar son solo de control, se suele usar *get*. Pero si son sensibles o para almacenarlos en una base de datos o en un fichero, es más aconsejable usar el método *post*. Con *get* existe un límite en el tamaño de los datos a enviar, en contra tiene la ventaja de que se permite almacenar en los marcadores el resultado.
- enctype = "application/x-www-form-urlencoded | multipart/form-data | text/plain".

  Si se usa el método *post*, se puede indicar 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 caracteres usarán la codificación URL antes de enviarlos.Con el valor "multipart/form-data" no se codifican los caracteres y sólo se usa cuando se envían archivos adjuntos al formulario. Con "text/plain" sólo los espacios en blanco se sustituyen por el signo "+", el resto de caracteres no se codifican.

Dentro de la etiqueta <form>, se colocan otras etiquetas llamadas controles de formulario. Es importante que estas etiquetas de formulario usen el atributo name para que los valores o datos asociados se envíen.

Las etiquetas de formulario son:

b) **<input>**. Sirve para enviar datos de diferente tipo. Es una etiqueta en línea con atributos básicos, de internacionalización, de eventos, y de foco. Tiene muchos otros atributos siendo los siguientes los más empleados:
   - type = "..." Especifica el tipo de elemento a visualizar. Los posibles valores más usados son:
     - text. Define un campo o cuadro de de texto para editar.
     - password. Define un campo de texto enmascarado (usa * para sustituir los caracteres).

- checkbox. Define una casilla de verificación.
- radio. Define un botón de radio.
- submit. Define 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. Define un campo de texto para teclear el nombre de un archivo y un botón para explorar en el sistema de archivos para seleccionar el archivo que se enviará al servidor.
- hidden. Define 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 *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" En los controles *text* y *password* pone el control en sólo lectura.
- src = "url", para indicar la ruta de la imagen del control *image* 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 a 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 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 o bien dicho control toma el foco.

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 del área.
- rows = "número". Número de filas del área.
- disabled = "disabled". Desactiva el <textarea> con lo que no se envía información.
- readonly = "readonly". De sólo lectura, por lo que no se puede escribir en el área.

g) **<button>** Representa un botón como los empleados mediante <input type="button">, pero permite a diferencia de éstos tener un contenido (texto, imágenes, etc.). Mediante el atributo type (valores button, submit y reset) se especifica el tipo de botón. Ejemplo:

```
<form action="demo_form.php" method="get">
        Elije una materia:
        <button name="materia" type="submit" value="mat_HTML">HTML</button>
        <button name="materia" type="submit" value="mat_CSS">CSS</button>
</form>
```

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

- size = "número". Fija el número de opciones visibles de la lista. Si es cero ó uno, la lista aparece sin desplegar; si es mayor que uno, la lista aparece desplegada.
- multiple = "multiple". Permite al usuario elegir más de una opción de la lista.

- disabled = "disabled", desactiva el control.

i) **<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 enviará al servidor.
- disabled = "disabled", desactiva esa opción de la lista.

j) **<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">VolksWagen</option>
    </optgroup>
</select>
```

Ejercicio: escribir el siguiente ejemplo y llamarlo prueba9.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <title>prueba9</title>
    </head>
    <body>
      <form action="#" method="get" enctype="text/plain">
            <label>Marcar las opciones que  os interesen:</label><br/>
            <fieldset>
                <legend>Agrupadas estan mas bonitas</legend>
                <input type="checkbox" />Opcion 1<br/>
                <input type="checkbox" />Opcion 2<br/>
                <input type="checkbox" />Opcion 3<br/>
            </fieldset>
      </form>
      <form action="#" method="get" enctype="text/plain">
            <label>Marcar las opciones que  os interesen:</label>
            <fieldset>
                <legend>Agrupadas estan mas bonitas</legend>
                <input type="checkbox" id="op1"/>
                <label for = "op1">Opcion 1</label><br/>
                <input type="checkbox" id="op2"/>
                <label for = "op2">Opcion 2</label><br/>
                <input type="checkbox" id="op3"/>
                <label for = "op3">Opcion 3</label><br/>
            </fieldset>
      </form>
    </body>
</html>
```

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.

Uno de ellos son los nuevos 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 dígitos. Su 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 obligado el 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): pattern="[A-Z]{5}[0-9]{3}"
- 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 desaparece en cuanto el cuadro de texto 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. Nuevas etiquetas semánticas de HTML5.

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 y cómo se estructura. El objetivo es indicar la semántica con HTML y dejar el formato exclusivamente para las hojas de estilo.

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

Unos de los problema del HTML en su versión 4.01 y anteriores es que no existían etiquetas capaces de agrupar bloques enteros y que tuviesen un significado por sí mismas. Por eso muchos desarrolladores terminaron usando la etiqueta como etiqueta para casi todo. En HTML5 se han incorporado una serie de etiquetas nuevas que permiten mejorar la semántica de nuestra página, etiquetas como <header>, <footer>, <nav>, <hgroup>,<aside>, <section> y <article>.

Esto no quiere decir que la etiqueta deje de usarse; sigue estando ahí y significando lo que siempre ha significado: conjunto de elementos. Se debe seguir usando como ayuda a la estructura de una página para crear el *layout* (esquema con la distribución de los elementos o plantilla) de ésta, 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 HTML/XHTML podría sustituirse por uno equivalente en HTML5 como se muestra a continuación:

```
<body>
    <div id="cabecera"> . . . </div>
    <div id="barraNavegacion">. . . </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 agrupar información introductoria como pueden ser el título o el logo. Realmente una página puede tener varios elementos <header>; por ejemplo, a nivel de la etiqueta <body> indica que su contenido es la cabecera de la página, pero dentro de <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: comúnmente 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 un estilo común. Si se utiliza dentro de <article> permite marcar su propia zona de títulos.
- **<aside/>** Permite indicar que ese bloque es solo un añadido a los bloques que tiene a su lado. Son datos extra, de forma que sin ellos podríamos pasar perfectamente pero que hemos decidido añadirlos al 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:

- Una cabecera conteniendo un logo y un menú de navegación principal.
- Posteriormente el contenido a leer, con una serie de datos de referencia sobre este contenido.
- Finalmente se crean dos bloques en dos columnas con contenidos en formato libre.

```
<body>
   <header>
      <h2>...</h2><a href="...">...</a>
      <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>
```

6. 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 un DTD, una Definición de Tipo de Documento.

La validación de un documento HTML no es estrictamente necesaria, de hecho hay millones de páginas de Internet que no son válidas, pero es bastante recomendable validar los documentos HTML ya que conseguiremos la seguridad necesaria para que otras aplicaciones o procesos sean más eficientes.

La validación podemos realizarla de dos maneras: en línea como es el caso del 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.

6.1. Validación en línea.

El consorcio W3C dispone de una herramienta de validación gratuita en Internet en la URL 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>
```

6.2. Herramientas de validación.

En el navegador *Firefox* podemos instalar un complemento para validar llamado *HTML Validator*. La URL para su descarga es https://addons.mozilla.org/es/firefox/addon/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 inferior derecha del navegador se mostrará un icono indicando si la página es válida o no. Si no lo es, aparecerá un icono en forma de cruz y colocando el puntero del ratón sobre él, se presenta un mensaje indicando el número de errores y advertencias. Haciendo doble clic se presentan todos los errores de forma detallada.

Para Chrome existe el complemento "Validity". Una vez instalado, aparecerá un icono a la derecha de la barra de direcciones, que al pulsarlo validará la página actual. El resultado de la validación se presenta en la Consola de las *Herramientas para desarrolladores*.

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*.