IFCT108PO

TECNOLOGÍA XML

IFCT108PO

TECNOLOGÍA XML

Alonso Rodríguez Zamora

Ra-Ma®

La ley prohíbe
fotocopiar este libro

IFCT108PO - TECNOLOGÍA XML

Editado por:
RA-MA Editorial
Calle Jarama, 3A, Polígono Industrial Igarsa
28860 PARACUELLOS DE JARAMA, Madrid
Teléfono: 91 658 42 80
Fax: 91 662 81 39
Correo electrónico: *editorial@ra-ma.com*
Internet: *www.ra-ma.es* y *www.ra-ma.com*
ISBN: 978-84-1897-100-6
Depósito legal: M-22400-2021
Maquetación: Antonio García Tomé
Diseño de portada: Antonio García Tomé
Filmación e impresión: Safekat
Impreso en España en julio de 2021

Dedicado con todo cariño a mi hijo Alonso

ÍNDICE

INTRODUCCIÓN

Publicación en Internet y tecnología XML es una obra dividida en dos partes tal y como su título indica:

1.ª parte, **Publicación en Internet:** un estudio sobre los principales lenguajes de marcas (HTML, DHTML, XHTML, WML) y otras tecnologías relacionadas con la publicación de documentos en la red como son las hojas de estilo CSS y lenguajes de guiones o *scripts* (JavaScript, VBScript). Se dan también unas nociones introductorias al lenguaje XML y DTD.

2.ª parte, **Tecnología XML:** se centra en las principales tecnologías relacionadas con XML como son los mecanismos de descripción de documentos (DTD y XML Schema) y hojas de estilo utilizadas para la publicación de documentos basados en XML: XSL-T y XSL-FO.

Para la comprensión del contenido de este libro no son necesarios conocimientos previos; en realidad, el objetivo de la primera parte es que sirva de introducción para que el lector diferencie perfectamente las características básicas de los diversos lenguajes y tecnologías utilizados para la publicación de documentos en Internet y principalmente (ya que es el tema central de la obra) que sepa crear documentos XML sencillos y publicarlos aplicando las técnicas CSS, es decir, que el lector esté preparado ya para entrar en la segunda parte de la obra en la cual se estudiará ya solamente la tecnología XML a fondo.

La primera parte de la obra es una introducción a la publicación de documentos en Internet; muestra de forma sencilla, clara y práctica cómo utilizar los lenguajes de marcas (HTML, DHTML, XML, etc.) y las tecnologías relacionadas (CSS, JavaScript, VBScript) para la publicación de documentos en Internet. Los capítulos dedicados a ello no pretenden ser un "curso de" ya que para cada uno de ellos se debería escribir un libro en concreto. El objetivo principal es mostrar el funcionamiento, las técnicas básicas y su aplicación práctica. Se pretende también que el lector diferencie claramente cada una de esas técnicas para, de esa manera, poder utilizar la que más le convenga en un determinado momento. Hemos de tener en cuenta que publicar en Internet no consiste en utilizar "un" lenguaje de marcas, sino el uso de diversas técnicas que se complementan.

Aunque, como he dicho, los capítulos de esta primera parte no son un estudio a fondo de los lenguajes tratados, y tampoco es el objetivo principal de este libro, los lectores cuentan con un magnífico complemento para ampliar las nociones expuestas; este complemento es el CD-ROM que acompaña al libro, el cual incluye tutoriales y herramientas básicas para quienes comienzan y para los lectores avanzados o aquéllos que quieran profundizar en un determinado tema numerosa documentación y utilidades relacionadas con el tratado en cada capítulo. Así, por ejemplo, como complemento al capítulo dedicado al lenguaje HTML, pensando en quienes se inician a este lenguaje, se incluye un tutorial de HTML y un editor; y, para quienes quieran profundizar, las Recomendaciones y DTD de este lenguaje, etc.

A lo largo de todos los capítulos de la primera parte se utilizará como documento de ejemplo un mismo texto, un pequeño artículo por llamarlo de alguna forma. El objetivo es que el lector experimente aplicando de forma práctica las técnicas expuestas y compruebe las posibilidades y limitaciones de cada una de ellas; comprenderá además por qué han de utilizarse en ciertos casos de forma conjunta para obtener unos resultados más satisfactorios.

La segunda parte se centra y profundiza en las tecnologías XML: el lenguaje XML, hojas de estilo XSL (XSL-T y XSL-FO), los mecanismos de descripción de documentos DTD y XML Schema. Se utilizará un documento XML de datos como ejemplo, una pequeña base de datos de usuarios con la cual el lector podrá experimentar y modificar o ampliar; al mismo tiempo aprenderá también los mecanismos utilizados para la publicación de esos documentos en Internet. El principal objetivo de esta parte es que el lector aprenda a manejar de forma práctica cada una de las tecnologías expuestas con ejemplos sencillos.

Para concluir, diré que Publicación en Internet y tecnología XML ***no es simplemente "un libro", sino una obra compuesta por un libro y diversos complementos incluidos en el CD-ROM que le acompaña***, el cual incluye numerosa documentación, estándares (recomendaciones) de lenguajes y tecnologías, herramientas y utilidades para efectuar las prácticas de cada materia tratada y, naturalmente, los códigos fuente de los ejemplos.

En realidad creo que es la obra que siempre me hubiera gustado haber tenido en mi biblioteca; quizás por eso me decidí a publicarla.

PUBLICACIÓN DE DOCUMENTOS EN INTERNET

Publicar un documento en Internet consiste en aplicar a unos determinados contenidos (texto, imágenes, etc.) una serie de formatos para que puedan ser visualizados desde una aplicación generalmente conocida con el nombre de navegador o explorador (*browser*) de Internet.

La aplicación de los formatos para la correcta visualización del documento se efectúa por medio de unas determinadas etiquetas (*tags*) definidas en los denominados lenguajes de marcas o de marcado. El lenguaje de marcado utilizado para dar formato a los documentos publicados en Internet, popularmente conocidas como páginas Web, es HTML en alguna de sus principales versiones estándar HTML3.2 y HTML4.0. Esta última está íntimamente relacionada con el denominado DHTML o HTML Dinámico. La más moderna versión del lenguaje HTML se conoce con el nombre de XHTML; este lenguaje utiliza las etiquetas propias de HTML y la normativa de XML. En realidad, XHTML es un lenguaje basado ya en XML.

XML es también un lenguaje de marcas, pero su finalidad no es dar formato a los documentos para su visualización, sino el establecer la estructura de los contenidos (datos) del documento. Para poder publicar documentos XML en Internet o, mejor dicho, poder visualizar su contenido en un navegador, se ha de recurrir a otras técnicas tales como las hojas de estilo en cascada (CSS) también utilizadas por HTML o tecnologías propiamente relacionadas con XML, tales como el lenguaje de estilos extensible (XSLT y XSL-FO). Debemos tener esto muy en cuenta ya que en este sentido podemos decir que XML no se asemeja en nada con HTML.

XML es un metalenguaje, es decir, un lenguaje que permite la creación de nuevos lenguajes. Existen ya diversos lenguajes basados en XML, como el ya citado XHTML. Otros son, por ejemplo: 3DML utilizado para la representación de escenas en tres dimensiones, CDF (Channel Definition Format o *Formato de Definición de Canales*) utilizado por el Servicio de Canales implementados en el navegador Microsoft Internet Explorer, MathML para la representación de fórmulas matemáticas, SMIL (Synchronized Multimedia Integration Language *o Lenguaje de Integración Sincronizada para contenidos Multimedia*), etc. Otro lenguaje basado en XML, y al cual dedicaremos un capítulo, es WML utilizado por la tecnología de telefonía sin hilos (WAP).

Todo lenguaje de marcas está definido en un documento denominado DTD (Document Type Definition o *Definición de Tipo de Documento*); en él se establecen las marcas, los elementos que son utilizados por dicho lenguaje y sus correspondientes etiquetas y atributos, su sintaxis y normas de uso. Si esto de las DTD le suena poco o,

incluso, nada, no debe preocuparse pues, tras la lectura de este libro, usted sabrá interpretar perfectamente esos documentos e, incluso, crear los suyos propios.

Además de los lenguajes de marcado y técnicas de aplicación de estilos, los documentos Internet pueden utilizar técnicas de programación basadas en lenguajes de Script, JavaScript y VBScript, principalmente para la manipulación del contenido del documento o, incluso, crear documentos totalmente nuevos de forma dinámica.

Para finalizar, podemos decir que el éxito de la publicación de documentos en Internet se basa principalmente en el conocimiento de las técnicas existentes (w3c) y de las implementaciones incorporadas en las aplicaciones visualizadoras (exploradores o navegadores).

W3C

World Wide Web Consortium, o W3C (http://www.w3.org), es el organismo dedicado a establecer los estándares, denominados Recomendaciones, de los lenguajes de marcas y otras tecnologías propias de Internet como son, por ejemplo, HTML, XML, XHTML, las hojas de estilo (CSS, XSL...), etc. Esta organización está formada por más de un centenar de empresas dedicadas a la informática y desarrollo de *software* relacionado con Internet: IBM, Sun Microsystems, Microsoft, Netscape Communications Corporation, Adobe, etc. Su forma de trabajar es por medio de publicaciones dadas a conocer en la Web de la organización, con los siguientes niveles de importancia (de menor a mayor):

- **Nota** propuesta realizada por una organización o grupo de organizaciones. Si tras un debate es considerada valiosa, es enviada a un Grupo de Trabajo para su discusión formal y comenzar el desarrollo del borrador de trabajo.

- **Borrador de trabajo** (Working Draft, WD) es el resultado del estudio realizado por el Grupo de Trabajo.

- **Propuesta de Recomendación** (Proposed Recommendation) son las especificaciones que se proponen a los miembros de la W3C para que sean aceptadas mediante votación. De ser aceptada, pasa al nivel superior y final.

- **Recomendación** (Recommendation) es ya un estándar que marca la normativa a seguir con los documentos de ese tipo.

LOS EXPLORADORES (*BROWSERS*) DE INTERNET

Una característica de los lenguajes de marcas, hojas de estilo, lenguajes de guiones o *scripts* y técnicas relacionadas es que se basan en la utilización de texto normal, por lo

cual un simple editor de texto podría ser suficiente para el desarrollo de aplicaciones basadas en estas tecnologías. Todos ellos son lenguajes interpretados que se ejecutan en la máquina cliente, por lo cual el único requisito para la correcta ejecución de la aplicación o visualización de un documento es tener instalado en nuestro equipo el intérprete adecuado.

En el caso de la publicación de documentos en Internet, la aplicación utilizada recibe el nombre de visualizador, visor, navegador o explorador (*browser*). Los de uso más extendido, y que han causado mayores polémicas también por sus continuos enfrentamientos, son Microsoft Internet Explorer y Netscape Navigator (o Communicator). Hemos de tener en cuenta que el desarrollador del explorador es el encargado de implementar el funcionamiento del intérprete de los lenguajes y tecnologías utilizadas en Internet basándose en las Recomendaciones propuestas por la W3C. De hecho, cada nueva versión de explorador supone la incorporación de un intérprete para una nueva tecnología; así, por ejemplo, los navegadores de versión 3 (navegadores de la tercera generación) estaban preparados para dar soporte al lenguaje HTML 3.2. Además, los citados navegadores implementaban algunas particularidades de la empresa creadora extra. Los navegadores de la versión 4 (cuarta generación) están preparados principalmente para dar soporte al denominado DHTML (Dynamic HTML) basado en el lenguaje HTML 4.0, uso de hojas de estilo, lenguajes de *script* y amplio soporte multimedia; naturalmente, cada desarrollador de *software* es el responsable de cómo diseñar y programar el intérprete utilizado para cada tecnología; por ello, la visualización de un mismo documento puede sufrir variaciones al ser abierto por uno u otro navegador.

Las últimas versiones de navegador incorporan algún módulo de *software* para dar soporte a alguna nueva tecnología. Microsoft Internet Explorer, en su versión 5, fue el primer navegador en incluir un "intérprete" (en este caso recibe el nombre de analizador o *parser*) para XML; por ello, para el desarrollo y prueba de los ejemplos de este libro propongo como navegador el uso de **Internet Explorer 5.0**, como mínimo, siendo más aconsejable la versión 5.5 y, si su equipo lo permite (Windows 98 mínimo), **Internet Explorer 6**, pues incorpora ya el procesador XML MSXML 3.0 (más adelante comprenderá para qué puede servir). En el ***CD-ROM directorio visores*** se incluye Microsoft Internet Explorer 5.5 y 6 en versión española para aquéllos que no dispongan de ellos. También se incluye una versión de Mozilla, otro de los pocos navegadores con soporte para XML.

Tras estos apuntes introductorios podemos comenzar ya el viaje por el fascinante mundo de la publicación de documentos en Internet. Deseo que la lectura de este libro sea de su agrado y disfrute haciendo sus primeros pinitos o le sirva para perfeccionar las técnicas que ya conozca.

Publicación en Internet

HTML
DHTML
CSS
XML
DTD
XSL
XHTML
WML

PARTE I

CAPÍTULO 1

EL DOCUMENTO

CONTENIDO, ESTRUCTURA Y FORMATO

Nuestra incursión en el mundo de la publicación comienza con un documento muy simple, un sencillo texto que hemos de colocar en un lugar de la red dedicado a la publicación de artículos sobre Internet.

Podemos decir que todo documento consta de los siguientes elementos: contenido, estructura y formato.

CONTENIDO DEL DOCUMENTO

El contenido (texto) de nuestro primer artículo, que será utilizado como elemento básico para las prácticas con las diversas tecnologías utilizadas en esta primera parte del libro, es el siguiente:

Lenguajes de marcas

SGML

Standard Generalized Markup Language o Lenguaje General Estándar de Marcado es el estándar para la creación y definición de otros lenguajes de marcado de documentos. SGML es utilizado para la creación de los denominados documentos de validación DTD (Document Type Definition o Definición de Tipo de Documento) en los cuales se definen los elementos, sintaxis y normas de un determinado lenguaje. SGML es, por lo tanto, un metalenguaje utilizado para la creación de otros lenguajes.

HTML

HiperText Markup Language o Lenguaje de Marcado de Hipertextos es el estándar utilizado para la aplicación de formatos y publicación de documentos en Internet. En HTML no hay separación entre contenido (datos) y presentación (formato), el formato se aplica directamente sobre el contenido. HTML es una aplicación SGML, un DTD, en el cual sus marcas (*tags* o etiquetas) están ya definidas.

XML

Extensible Markup Language o Lenguaje de Marcado Extensible. XML se basa en SGML y es, por así decirlo, una versión reducida de él. Al igual que SGML, es un lenguaje orientado a la definición de la estructura de los documentos pero no a su representación (aplicación de formatos). XML es también un metalenguaje. El lenguaje XML no tiene etiquetas predefinidas.

XHTML

Extensive HiperText Markup Language o Lenguaje de Marcado de Hipertextos Extensible. Es un lenguaje basado en XML; podemos decir que es el lenguaje HTML pero aplicando la normativa utilizada por XML. XHTML está orientado a la creación de documentos bien estructurados y aplicación de formatos.

Sábado, 4 de enero de 2003

Alonso Rodríguez Zamora

ESTRUCTURA DEL DOCUMENTO

La estructura viene establecida por la existencia en el documento de una serie de elementos claramente diferenciados unos de otros. En nuestro caso podríamos decir que los elementos relevantes de los cuales se compone el artículo son:

- **título**: Lenguajes de marcas.
- **secciones (4):** que, a su vez, están formadas por dos elementos:
  - **títulos de sección (4)**: SGML, XML, XHTML, HTML.
  - **párrafos (4)**: explicación de cada uno de los lenguajes.
- **fecha:** sábado, 4 de enero de 2003.
- **autor:** Alonso Rodríguez Zamora.

La estructura del documento está definida por la forma en como se distribuyen los elementos que lo componen. En dicha estructura cada elemento es una unidad independiente que no puede interferir con otro elemento. Las estructuras de los documentos se pueden representar gráficamente como un esquema en forma de árbol; el correspondiente a nuestro ejemplo puede verlo en la Figura 1 - 1.

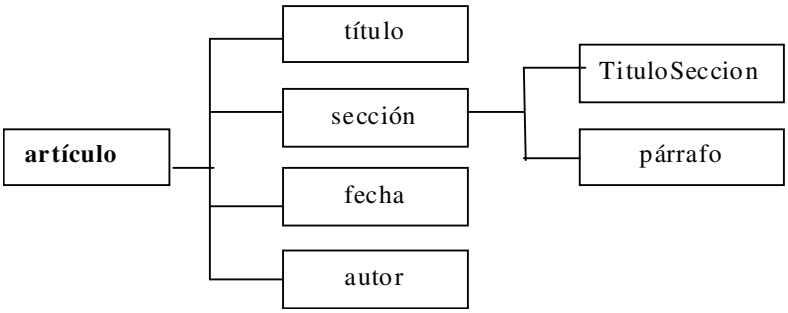

Figura 1 - 1. Representación gráfica de la estructura del documento

Concluiremos este apartado afirmando que todo documento, además del contenido (texto o datos), y por muy simple que sea, tiene una determinada estructura en la cual existen una serie de elementos claramente diferenciados y que, además, pueden ser representados gráficamente mediante un esquema en forma de árbol.

Quizás se esté preguntando a qué viene todo esto sobre contenidos y estructuras; de ser así le adelantaré que XML está especialmente diseñado para trabajar con las estructuras y contenidos de los documentos; por lo tanto, debe empezar a familiarizarse con estos términos.

FORMATO DEL DOCUMENTO

El artículo en el momento actual es un documento sin formato, lo que generalmente se conoce como texto plano. Por lo general, en todo tipo de publicación impresa tras la escritura del documento se pasa a darle formato, es decir, utilizando un procesador de texto o cualquier otra herramienta adecuada, se marcan y diferencian los diversos elementos de que consta: título, subtítulos, párrafos, se establecen los márgenes, sangrías, etc. Para dar formato al documento, se utilizan unos determinados estilos: tipos de fuente de letra, color, tamaño...; de esta forma, el documento queda más atractivo y su lectura se hace más fácil.

Nuestro documento, tras utilizar un procesador de textos y aplicar los correspondientes estilos, presenta el siguiente aspecto:

Lenguajes de marcas

SGML

Standard Generalized Markup Language o ***Lenguaje General Estándar de Marcado*** es el estándar para la creación y definición de otros lenguajes de marcado de documentos. SGML es utilizado para la creación de los denominados documentos de validación **DTD** (**Document Type Definition** o ***Definición de Tipo de Documento***) en los cuales se definen los elementos, sintaxis y normas de un determinado lenguaje. SGML es, por lo tanto, un metalenguaje utilizado para la creación de otros lenguajes.

HTML

HiperText Markup Language o ***Lenguaje de Marcado de Hipertextos*** es el estándar utilizado para la aplicación de formatos y publicación de documentos en Internet. En HTML no hay separación entre contenido (datos) y presentación (formato); el formato se aplica directamente sobre el contenido. HTML es una aplicación SGML, un DTD, en el cual sus marcas (*tags* o etiquetas) están ya definidas.

XML

Extensible Markup Language o ***Lenguaje de Marcado Extensible***. XML se basa en SGML y es, por así decirlo, una versión reducida de él. Al igual que SGML, es un lenguaje orientado a la definición de la estructura de los documentos pero no a su representación (aplicación de formatos). XML es también un metalenguaje. El lenguaje XML no tiene etiquetas predefinidas.

XHTML

Extensive HiperText Markup Language o ***Lenguaje de Marcado de Hipertextos Extensible***. Es un lenguaje basado en XML, podemos decir que es el lenguaje HTML pero aplicando la normativa utilizada por XML. XHTML está orientado a la creación de documentos bien estructurados y aplicación de formatos.

Sábado, 4 de enero de 2003

Alonso Rodríguez Zamora

ESTILOS

Los estilos consisten en la definición de determinadas características utilizadas por el texto, tales como fuente de letra, tamaño, color, etc. En el documento del ejemplo los estilos utilizados son los siguientes:

- **Título 1**

Fuente: Comic Sans MS
Tamaño: 16
Color: rojo
Alineación: centro
Negrita

- **Título 2**

Fuente: Comic Sans MS
Tamaño: 14
Color: azul
Alineación: izquierda
Subrayado

- **Párrafos**

Fuente: Times New Roman
Tamaño: 11
Color: negro
Sangrado primera línea: 15
Alineación: izquierda
Normal

- **Fecha**

Fuente: Arial
Tamaño: 10
Color: negro
Alineación: izquierda
Normal

- **Autor:**

Fuente: Arial
Tamaño: 10
Color: negro
Sangrado: 6
Alineación: izquierda
Itálica

Podemos observar que, con la aplicación de los estilos, en realidad lo que estamos haciendo es diferenciar los diversos elementos que componen la estructura del documento, además de proporcionar una mejor visualización y lectura.

Cuando se trabaja con un procesador de texto, los formatos se aplican de forma automática, los mecanismos utilizados quedan ocultos, pero sabemos que se están aplicando de alguna forma una serie de marcas que afectarán a la presentación del documento, aunque, utilizando técnicas diferentes, la publicación de documentos en Internet se basa también en aplicar formatos a los diversos elementos que componen el documento utilizando para ello un lenguaje de marcas.

Por lo general, los procesadores de texto avanzados permiten guardar un mismo documento en diversos tipos de archivo para que puedan ser utilizados por determinadas aplicaciones. Según el tipo seleccionado, los formatos se aplican y codifican de forma distinta pudiendo quedar ocultos al usuario o ser totalmente transparentes como sucede en el caso de los lenguajes de marcas utilizados para la publicación de documentos en Internet. En estos último las etiquetas (sirven para definir los formatos) y el contenido se guardan en texto plano. Entre las posibilidades "Guardar como..." los procesadores de texto suelen permitir tipos de archivo RTF (Rich Text Format o Formato de Texto Enriquecido) o, incluso, HTML (HiperText Markup Language o Lenguaje de Marcado para HiperTextos), el lenguaje estándar utilizado mayoritariamente para la publicación en Internet. He utilizado estas dos opciones tras haber aplicado los formatos al artículo; se incluyen en el ***CD-ROM directorio Ejemplos\Cap01,*** junto con el texto plano (articulo.txt).

Puede abrir el archivo artículo.rtf con casi cualquier procesador de textos por sencillo que sea; podrá tener información de los estilos aplicados colocando el cursor sobre el texto; la barra de herramientas de formato del procesador le indicará cuáles son los aplicados para cada parte del documento, pero no podrá observar cómo se aplican esos formatos.

Para visualizar el archivo articulo.htm, necesita tener instalado un explorador de Internet; basta con que haga un doble clic sobre él desde el explorador de Windows para que pueda acceder a él; de no ser así, deberá instalar un explorador (el incluido en el CD-ROM, por ejemplo). Una vez abierto, podrá observar la forma en como se han aplicado los formatos desde la opción "Ver → Código fuente" (Microsoft Internet Explorer) o alguna similar dependiendo del navegador utilizado. Comprobará que en la ventana de visualización del código, el Bloc de Notas en realidad, junto con el texto del artículo se mezclan elementos extraños encerrados entre los símbolos (marcas) menor que (<) y mayor que (>), como la siguiente línea, por ejemplo:

```
<U><FONT FACE="Comic Sans MS" SIZE=4 COLOR="#0000ff"> <P> SGML </P>
</U></FONT>
```

Estos "raros" elementos son los correspondientes a las etiquetas del lenguaje HTML utilizadas para dar formato al documento y se representan tal y como los visualiza en el explorador.

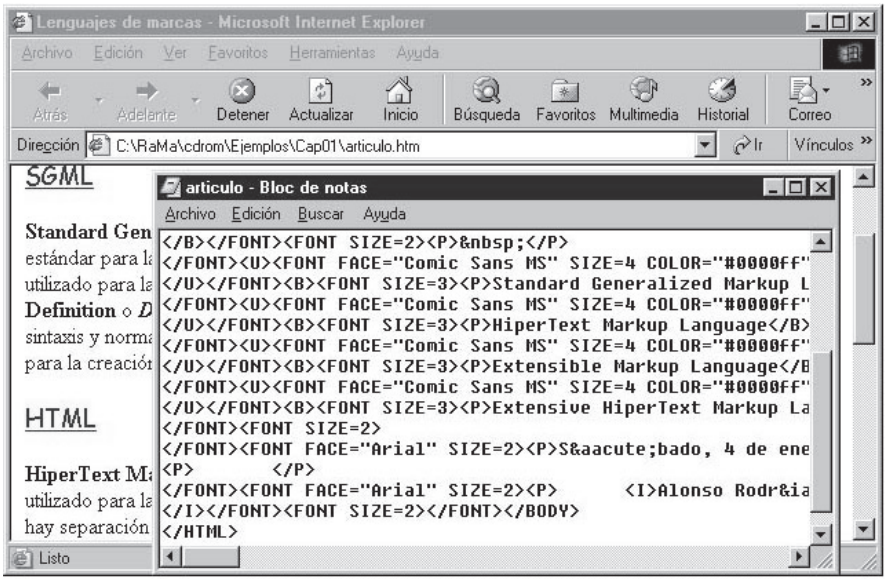

Figura 1 - 2. Visualización del documento y etiquetas de formato HTML

Si observa los dos documentos, articulo.pdf y artículo.htm, podrá comprobar que existen diferencias entre ambos, y ninguno de ellos mantiene el formato original tal como podemos ver en el documento modelo propuesto. Todos ellos se parecen, pero su formato es distinto...; observe, por ejemplo, la falta de sangría de primera línea de los párrafos en el documento HTML generado. Nuestro objetivo, por lo tanto, consistirá en dar formato al documento para que sea publicado en Internet y se visualice de forma idéntica al modelo propuesto. Utilizaremos para ello todas las técnicas existentes: HTML, XML, CSS, etc.; no es que seamos maniáticos, es un reto (y además una magnífica excusa para comenzar nuestro viaje por las tecnologías utilizadas en Internet para la publicación de documentos).

Allá vamos...

CAPÍTULO 2

PUBLICACIÓN CON HTML

INTRODUCCIÓN

El presente capítulo pretende cubrir los siguientes objetivos:

1. Enseñar al lector principiante, de una forma sencilla y clara, cómo se utilizan los lenguajes de marcas para dar formato a los documentos que se quieren publicar en Internet.

2. Proporcionar los conocimientos básicos para poder comenzar a practicar o experimentar por su cuenta con el lenguaje HTML. En el CD-ROM se incluye diverso material complementario con el cual ampliar las nociones básicas. Entre estos complementos podrá encontrar un tutorial de HTML 3.2, un editor y documentación utilizada por los profesionales de desarrollo Web.

3. Efectuar un análisis del denominado HTML estático, HTML 3.2, con sus posibilidades y limitaciones; de esta forma comprenderemos por qué HTML necesita apoyarse en tecnologías complementarias.

4. Para quienes ya tengan conocimientos de HTML, pero desconozcan XML y la utilización de las DTD (Document Type Definition o Definición de Tipo de Documento), el objetivo principal es mostrar cómo se definen los lenguajes de marcas, es decir, comenzar a tomar contacto con las DTD que, aunque generalmente ignoradas, cuando se publica con HTML, estarán casi siempre presentes en la programación con XML.

HTML

HiperText Markup Language o ***Lenguaje de Marcado de Hipertextos*** es en la actualidad el lenguaje más utilizado para creación de las denominadas páginas Web y la publicación de documentos en Internet. HTML es el lenguaje estándar utilizado para la aplicación de formatos a los documentos. En HTML no hay separación entre contenido (datos o texto) y presentación (maquetación y formato) ya que el formato se aplica directamente sobre el contenido. También se dice que HTML es una aplicación SGML (Standard Generalized Markup Language o Lenguaje de Marcado de uso Generalizado Estándar) ya que utiliza declaraciones SGML y definiciones de tipo de documento (DTD).

Como su nombre indica, el principal objetivo del lenguaje HTML es el permitir a diversos documentos de texto comunicarse entre ellos por medio de hiperenlaces (HiperText) a nivel local o remoto. Además, incorpora elementos para la maquetación y aplicación de formatos y estilos a esos documentos.

Existen varias Recomendaciones de la W3C para este lenguaje correspondientes a sus diversas versiones; no obstante, podemos decir que los auténticos estándares son los establecidos en las Recomendaciones de HTML 3.2 y HTML4.0

- ***Las Recomendaciones de HTML3.2*** representan el HTML básico, el correspondiente al denominado HTML estático y diseñado principalmente para dar formato a los contenidos y permitir la interrelación entre documentos. Se puede decir que la única interactividad existente con el usuario es la proporcionada por esos hiperenlaces, es decir, la posibilidad de acceder a otros documentos existentes en cualquier lugar de la red.

- ***Las Recomendaciones de HTML 4.0*** introdujeron muchas novedades; es el estándar utilizado por el denominado DHTML (Dynamic HTML o HTML Dinámico) y, entre sus principales características, está la utilización de nuevas etiquetas y nuevos atributos destinados a proporcionar soporte a las hojas de estilo en cascada (CSS), a los lenguajes de *script*, la posibilidad de poder incluir programas externos (*applets*, controles activeX, etc.), un amplio soporte multimedia y acceso a bases de datos. Como conclusión, podemos decir que aparecen etiquetas nuevas y se incorpora la utilización de atributos comunes y gestión de eventos. Los elementos HTML son tratados como objetos, con propiedades y métodos; por ello, pueden ser programados pudiendo modificar su aspecto y comportamiento mientras se visualiza la página, es decir, son objetos dinámicos.

Este capítulo es una introducción a los lenguajes de marcas y al lenguaje HTML; para ello utilizaremos las Recomendaciones de HTML 3.2; esto nos permitirá conocer sus posibilidades y limitaciones.

HTML 3.2 O HTML ESTÁTICO

Como ya hemos dicho, el denominado HTML estático se basa en las Recomendaciones de HTML 3.2. Una Recomendación es un documento publicado en la sede de la W3C en el cual se establece la normativa, sintaxis y elementos utilizados por un determinado lenguaje.

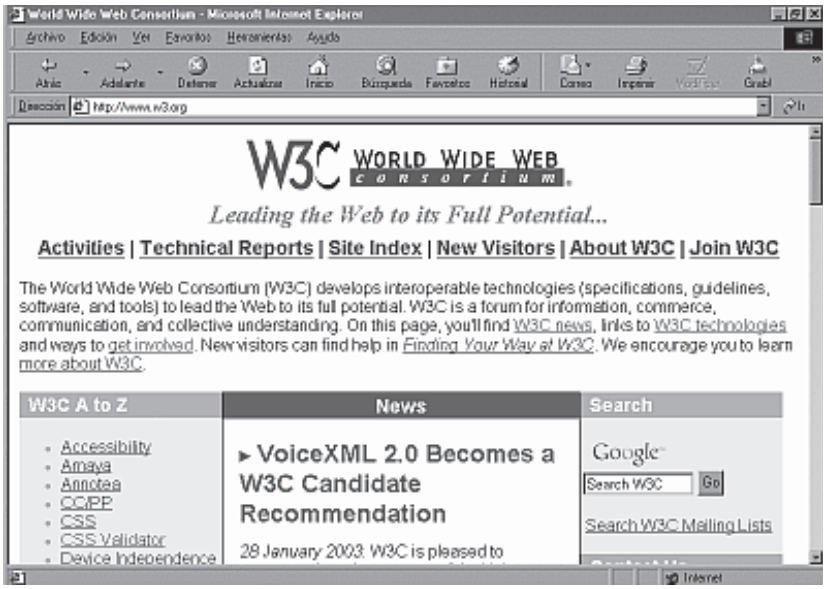

Figura 2 - 1. Página principal del W3C (http://www.w3.org) en febrero de 2003. Visita obligada para todos los interesados en la publicación de documentos en Internet

Una recomendación es en realidad una DTD por medio de la cual ese lenguaje se convierte en un estándar. El lenguaje HTML 3.2 se encuentra definido en el documento "***Especificaciones de referencia de HTML3.2 Recomendación del W3C 14-ene-1997***".

REC-html32

HTML 3.2 Reference Specification W3C Recommendation 14-Jan-1997

http://www.w3.org/TR/REC-html32

En este documento se establece la normativa sobre la estructura de los documentos HTML 3.2, los elementos que componen este lenguajes con sus etiquetas y atributos y las entidades utilizadas. Además incluye la Definición de Tipo de Documento (DTD), las entidades de caracteres para ISO Latin-1 y la tabla de códigos de caracteres imprimibles de Latin-1. Se puede obtener en la dirección de Internet arriba indicada; no obstante, pensando en quienes no tengan la posibilidad de acceder a él por este medio, ha sido incluido en el ***CD-ROM directorio html\html32\en***. Existe también una versión en

castellano en *http://dns.uncor.edu/info/html/rec-sp.htm,* también incluida en el ***CD-ROM directorio html\html32\sp***.

ESTRUCTURA DE UN DOCUMENTO HTML

Los elementos que componen el lenguaje HTML se representan dentro de un documento HTML por medio de marcas denominadas etiquetas. La estructura básica de un documento HTML está representada por los siguientes elementos:

- **HTML**: establece el documento basado en este lenguaje. Está representado por las marcas **<HTML>** y **</HTML>**, las cuales sirven para indicar el inicio y final del documento, respectivamente.

- **HEAD**: sección de cabecera del documento; su contenido son una serie de etiquetas especiales con información o metadatos complementarios al documento. Está representado por las marcas **<HEAD>** y **</HEAD>**. Dentro de la sección de cabecera es obligatorio definir el título del documento por medio del elemento **TITLE**, etiquetas **<TITLE></TITLE>**.

- **BODY**: sección de cuerpo del documento; su contenido es el documento en sí y está compuesto por las etiquetas utilizadas para dar formato y el texto, marcas **<BODY>** y **</BODY>**.

Podemos decir que en la práctica la estructura básica de un documento HTML muestra el siguiente aspecto:

```
<HTML>
   <HEAD>
      <TITLE>Titulo del documento</TITLE>
      ... contenido de la sección de cabecera (metainformación) ...
   </HEAD>
   <BODY>
      ... contenido de la sección del cuerpo del documento ...
   </BODY>
</HTML>
```

Como puede observar, las etiquetas que representan a un determinado elemento en HTML suelen ir por parejas: existe una etiqueta para marcar el inicio del elemento y otra para indicar su final; entre ambas se sitúa el **contenido**. El contenido puede estar formado por otros elementos HTML (otras etiquetas) o texto. Los elementos HEAD y BODY son

contenido del elemento HTML (también denominado elemento raíz del documento) y sirven para delimitar las dos secciones de las cuales consta todo documento HTML.

El nombre de las etiquetas inicial y final es el mismo: el correspondiente al elemento va entre las marcas menor que ("<") y mayor que (">"). La etiqueta de cierre utiliza además delante del nombre del elemento el símbolo de barra inclinada ("/"). La marca representativa de las etiquetas de cierre es, por lo tanto, "</". No deben existir espacios entre las marcas de las etiquetas y su nombre. La sintaxis genérica para el uso de las etiquetas es la siguiente:

<ETIQUETA> contenido </ETIQUETA>

A continuación se muestra un breve pero interesante fragmento de las Recomendaciones HTML 3.2; es el referente a la Estructura de un documento basado en HTML3.2 y dice lo siguiente:

Los documentos HTML 3.2 comienzan con una declaración <!DOCTYPE> seguidas por un elemento HTML conteniendo un elemento HEAD y luego un elemento BODY:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>Un estudio de la dinámica de poblaciones</TITLE>
 ... otros elementos del encabezamiento
</HEAD>
<BODY>
 ... cuerpo del documento
</BODY>
</HTML>
```

En la práctica, las tareas de comienzo y final HTML, HEAD y BODY pueden ser omitidas de la indicación de formato al poder las mismas ser inferidas en todos los casos por intérpretes que se adecuan a la DTD del HTML 3.2.

Todo documento que satisfaga el HTML 3.2 debe comenzar con la declaración <!DOCTYPE> que es necesaria para distinguir a los documentos HTML 3.2 de otras versiones del HTML....

Según este texto debemos poner atención a los siguientes puntos:

1.º Un documento HTML 3.2 debe comenzar obligatoriamente por una declaración como la siguiente:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">

Esa línea es lo que generalmente se conoce como línea de prólogo de un documento; en ella se establece qué tipo de documento es y la DTD del lenguaje utilizado y su versión, es decir:

- DOCTYPE HTML establece que es un documento de tipo HTML.

- PUBLIC "-//W3C//DTD HTML 3.2 Final//EN" indica que está basado en la DTD de HTML 3.2 Final y que está en lengua inglesa.

En la práctica, y de forma casi generalizada, esta línea ha sido omitida o apenas tenida en cuenta por los diseñadores de páginas Web. Comienza a tener importancia con la llegada de XML y la validación de documentos. Más adelante volveremos a tratar el tema de la validación más ampliamente.

2.º "Las tareas de comienzo y final HTML, HEAD y BODY pueden ser omitidas".

Este punto es muy importante; en él se establece que las marcas más relevantes, que son las que establecen la estructura básica del documento indicando el comienzo del mismo (nodo raíz) y las que determinan el bloque de cabecera (HEAD) y el bloque del cuerpo del documento (BODY), pueden ser omitidas por el programador de la página, pues se supone que serán 'inferidas' por el intérprete (incorporado en el explorador) utilizado.

Ésta es una simple muestra de lo permisivo que puede llegar a ser HTML. La "permisividad" es una característica de HTML y establece otra gran diferencia con XML, que es un lenguaje muy estricto. Otra característica de este lenguaje es la existencia de elementos que no requieren el uso de la etiqueta final; el ejemplo más representativo lo encontramos en los párrafos (marca <P>), pues se supone que, cuando un párrafo comienza, es que otro ha finalizado..., es decir, se puede utilizar de forma indistinta:

```
<P>Párrafo de texto....
<P>Párrafo de texto... </P>
```

También son permitidos los solapamientos de marcas, es decir, si queremos poner un párrafo en negrita (marca <B>) e itálica (marca <I>), no importa el que el anidamiento de las marcas sea incorrecto, es decir, el intérprete nos mostraría el mismo resultado poniendo lo siguiente:

```
<B><I> párrafo en negrita e itálica </B></I>
<I><B> párrafo en negrita e itálica </I></B>
<B><I> párrafo en negrita e itálica </I></B>
<I><B> párrafo en negrita e itálica </B></I>
```

En HTML las cuatro formas pueden funcionar ya que el intérprete las dará como correctas. En XML solamente las dos últimas son consideradas sintácticamente correctas y el analizador (*parser*) nos informaría de la incorrección cometida en las dos primeras y la carga del documento se interrumpiría. Más adelante hablaremos de los "analizadores" de XML; de momento le diré que sirven para comprobar si la estructura de un documento es correcta; de no ser así, no existe la posibilidad de acceder a su contenido.

HTML es un lenguaje "interpretado"; el intérprete es un programa que se dedica a leer las marcas HTML y aplicar el formato indicado por la marca utilizada. Los desarrolladores de los intérpretes, por lo tanto, son los que establecen en la implementación del código cómo se debe de interpretar esa etiqueta, cómo se ha de mostrar su contenido en el documento. Se supone que esas implementaciones han de seguir naturalmente las recomendaciones establecidas por el W3C. Los navegadores, además, suelen incluir la posibilidad de utilizar etiquetas que no existen en el estándar y que solamente funcionan en ese navegador. Por otra parte, el navegador puede dar soporte a unas determinadas tecnologías en su totalidad o parte de ella. Teniendo todo esto en cuenta, no debemos extrañarnos de que, a veces, la visualización de un mismo documento HTML en un navegador u otro pueda variar, el rango de la variación puede ir de lo insignificante a lo asombroso. Como resultado de todo ello, tenemos que, a veces, los desarrolladores de páginas Web se ven obligados a hacer, como mínimo, dos versiones de un mismo lugar si quieren llegar a todos los usuarios. Estas dos versiones se corresponden naturalmente con las de los navegadores más extendidos: Microsoft Internet Explorer y Netscape Navigator.

ELEMENTOS Y ETIQUETAS DE HTML

Como ya hemos dicho, los elementos del lenguaje HTML son representados en un documento por medio de sus correspondientes etiquetas. Los nombres de las etiquetas HTML pueden ir en mayúsculas, minúsculas o combinadas, es decir, es lo mismo utilizar <H1>Título principal</H1> que <h1>Título principal</h1> o <h1>Título principal</H1>. El contenido se mostrará igualmente con cualquiera de las etiquetas utilizadas. Con los documentos basados en HTML se recomienda el uso de caracteres en mayúscula pues en la DTD están definidos de esta forma; los basados en XHTML, en cambio, utilizan las etiquetas en minúscula.

Según lo visto hasta ahora, las etiquetas sirven para delimitar determinados bloques de información dentro del documento y, por ello, suelen ir por parejas. No obstante, en HTML algunos elementos pueden utilizar solamente la etiqueta inicial y omitirse la final interpretándose entonces que el bloque iniciado finaliza cuando aparezca otra etiqueta de inicio de cualquier otro elemento. En ambos casos estas etiquetas utilizan un determinado contenido que puede ser otros elementos HTML o texto.

Existen también elementos vacíos, sin contenido, que, como es lógico, utilizan siempre una sola etiqueta ya que no sirven para aplicar formato a ningún contenido.

Podríamos, por lo tanto, establecer los siguientes grupos de elementos y etiquetas utilizadas:

Elementos con contenido

Como su nombre indica, se utilizan como contenedor de otros elementos de HTML o de texto. Este tipo de elementos en HTML pueden usar:

• **Doble etiqueta**: han de utilizar obligatoriamente dos etiquetas para indicar dónde comienza (etiqueta de inicio) y dónde termina (etiqueta de cierre) la sección (contenido) sobre la cual se aplica el formato. Entre éstas tenemos las utilizadas para establecer la apariencia de los caracteres: negrita (pareja <B> </B>), itálica (pareja <I> </I>), etc.

Su sintaxis genérica es: <ETIQUETA>contenido</ETIQUETA>.

Ejemplo: <H1>Título principal</H1>

• **Etiqueta simple**: se puede usar solamente la etiqueta de inicio de la sección sobre la cual se aplica la marca o el formato. Se corresponden con aquellas etiquetas que en el DTD utilizan - O en la declaración del elemento, como es el caso de los elementos de párrafo <P>, los ítem de listas <LI>, etc.

Su sintaxis genérica es: <ETIQUETA> Contenido.....

Ejemplo: <P> párrafo...

Elementos sin contenido (elementos vacíos)

No se utilizan sobre ningún contenido, son aquellos elementos que en la DTD están declarados como EMPTY. Naturalmente, están representados por una sola etiqueta.

Uso: <ETIQUETA>

Ejemplos: <HR> (traza una línea horizontal),
 (efectúa un avance de línea).

Las etiquetas de HTML pueden utilizar una serie de parámetros adicionales conocidos con el nombre de atributos. Uno de estos atributos, utilizado por diversas etiquetas de tipo bloque, es ALIGN, que sirve para establecer cómo se alineará el elemento que lo utiliza en el documento. Sus posibles valores son *left* (izquierda), *center* (centro) o *right* (derecha). Así, por ejemplo, si quisiéramos poner un título centrado, utilizaríamos la siguiente marca:

```
<H1 ALIGN="CENTER">Título centrado</H1>
```

BREVE REFERENCIA DE HTML 3.2

A continuación se ofrece una pequeña referencia de los elementos utilizados por HTML3.2; no se indican los atributos, simplemente se muestra tal y como hemos visto en el apartado anterior si han de utilizar dos etiquetas o una y, de ser así, si el elemento se aplica sobre un determinado contenido, o bien, es un elemento vacío. Los elementos del lenguaje HTML se pueden dividir en tres grandes grupos:

- **Elementos de estructura básica del documento**: el elemento documento HTML y las secciones HEAD y BODY ya comentados anteriormente. No debemos olvidar al elemento que da título al documento, TITLE, en la sección de cabecera.

- **Elementos de nivel bloque (*block*)**, utilizados como contenedor de otros elementos de nivel bloque, de nivel texto y datos de caracteres (texto) específicos existentes en un documento. Su principal característica es que efectúan una ruptura (salto de línea) dentro de él diferenciándose por ello claramente unos de otros en el documento.

- **Elementos de nivel texto (*inline*),** utilizados generalmente como contenido de los elementos de nivel bloque para aplicar un estilo determinado al texto sobre el cual se aplican.

Elementos de estructura básica del documento

HTML - Documento HTML, etiquetas:	<HTML> </HTML>
HEAD - Cabeza del Documento, etiquetas:	<HEAD> </HEAD>
BODY - Cuerpo del Documento, etiquetas:	<BODY> </BODY>

Elementos de la sección de cabecera (*head*)

A) contenedores

TITLE - Título del Documento	<TITLE> #PCDATA </TITLE>
SCRIPT - *Script* incrustado	<SCRIPT> CDATA </SCRIPT>
STYLE - Estilo incrustada	< STYLE> CDATA </STYLE>

#PCDATA (Parser Character Data o Datos de Caracteres analizados) indica que el contenido textual de este elemento puede ser analizado por el intérprete de HTML; de

esta forma se pueden incluir, por lo tanto, referencias a entidades utilizadas para los caracteres de Latin-1 y especiales entre los símbolos & (*ampersand*) y ; (punto y coma).

CDATA (Character Data o Datos de Caracteres) indica que el contenido textual será enviado como datos de caracteres que no han de ser analizados por el intérprete de HTML; por ello, no se deben utilizar caracteres especiales ni entidades de Latin-1. En los elementos SCRIPT y STYLE este tipo de contenido debe ir situado entre las marcas de comentarios <!-- y --> para que sea omitido por el intérprete ya que, de no ser así, el texto aparecería como tal en el documento.

B) sin contenido

BASE - URI Base del Documento	<BASE> (vacío)
ISINDEX - Prompt de entrada de datos	<ISINDEX> (vacío)
LINK - Enlaces del Documento	<LINK> (vacío)
META - Metadatos	<META> (vacío)

Elementos de nivel bloque (*block*)

Generalmente son utilizados como contenedores, aunque también pueden ser contenido de otro elemento de nivel bloque. Pueden contener otro elemento de nivel bloque, elementos de nivel texto y texto (#PCDATA).

ADDRESS - Dirección	<ADDRESS> </ADDRESS>
BLOCKQUOTE - Cita	<BLOCKQUOTE> </BLOCKQUOTE>
CENTER - Bloque Centrado	<CENTER> </CENTER>
DIV - División de párrafo	<DIV> </DIV>
H1 - Título de nivel uno	<H1></H1>
H2 - Título de nivel dos	<H2></H2>
H3 - Título de nivel tres	<H3></H3>
H4 - Título de nivel cuatro	<H4></H4>
H5 - Título de nivel cinco	<H5></H5>

H6 - Título de nivel seis	<H6></H6>
HR - Línea horizontal	<HR> (vacío)
PRE - Texto Preformateado	<PRE> </PRE>
P - Párrafo	<P> ... contenido...

Formularios

FORM - Formulario	<FORM></FORM>

Tablas

TABLE - Tabla	<TABLE></TABLE>

Listas

OL - Lista ordenada (Ordered List)	<OL></OL>
UL - Lista desordenada (Unordered List)	<UL></UL>
LI - Ítem de Lista (List Item)	<LI>contenido
DL - Lista de Definiciones (Definition List)	<DL></DL>
DT - Término de la Definición (Definition Term)	<DT> contenido
DD - Descripción de la Definición (Definition Description)	<DD> contenido
DIR - Lista de Directorio (Directory)	<DIR> </DIR>
MENU - Lista de Menú	<MENU></MENU>

Elementos de nivel texto (*inline*)

Los elementos de nivel texto pueden tener como contenido otros elementos de nivel texto y texto (#PCDATA). Son utilizados como contenido de los elementos de nivel bloque ya que están especialmente diseñados para aplicar formatos y estilos característicos (poner en negrita, subrayar, etc.) a determinadas partes de un párrafo o caracteres en concreto. Su principal característica es que no efectúan ruptura de línea, como hacen los elementos de nivel bloque; por ello, se les conoce también como elementos en línea o *inline*.

Elementos de estilo de fuente (elementos de estilo tipográfico)

B - Texto en negrita (Bold)	<B></B>
BIG - Texto grande	<BIG></BIG>
I - Texto en Itálica	<I></I>
S - Texto tachado (Strike-through)	<S></S>
SMALL - Texto pequeño	<SMALL></SMALL>
STRIKE - Texto tachado (Strike-through)	<STRIKE></STRIKE>
TT - Texto de Teletipo	<TT></TT>
U - Texto subrayado (Underlined)	<U></U>

Elementos de frase, oración

CITE - Cita	<CITE></CITE>
CODE - Código de Computadora	<CODE></CODE>
DFN - Definición	<DFN></DFN>
EM - Énfasis	<EM></EM>
KBD - Texto de teclado	<KBD></KBD>
SAMP - Ejemplo	<SAMP></SAMP>
STRONG - Énfasis	<STRONG></STRONG>
VAR - Variable	<VAR></VAR>

Elementos de campos de formulario (FORM)

INPUT - Entrada de datos	<INPUT> (vacío)
SELECT - Selector de Opciones	<SELECT> </SELECT>
OPTION - Opción de Menú	<OPTION> (vacío)
TEXTAREA - Texto multilínea	<TEXTAREA></TEXTAREA>

Elementos de tablas (TABLE)

CAPTION - Título de la Tabla	<CAPTION></CAPTION>
TR - Fila de la Tabla (Table Row)	<TR>... contenido...
TD - Datos de la Tabla (Table Data)	<TD>... contenido...
TH - Títulos de la Tabla (Table Header)	<TH>... contenido...

Elementos especiales

A - Ancla	<A> </A>
APPLET - applet Java	<APPLET> </APPLET>
PARAM - Parámetros de Applet	<PARAM> (vacío)
BASEFONT - Fuente Base	<BASEFONT></BASEFONT>
BR - Ruptura de línea	 (vacío)
FONT - Fuente	<FONT></FONT>
IMG - Imagen	< IMG> (vacío)
MAP - Mapa de Imagen	<MAP></MAP>
AREA - Región de un Mapa de Imagen	<AREA> (vacío)

Debido a las características y objetivos de esta obra, no nos extenderemos en las páginas del presente capítulo con mucha más información sobre HTML 3.2, aunque hablaremos un poco más de los atributos. No obstante, si no conoce este lenguaje y quiere aprender más sobre él, no tiene por qué sentirse defraudado ya que, tal como se indicaba en la introducción, esta obra cuenta con una serie de **complementos** a los temas tratados en los capítulos del libro. Estos complementos se incluyen en el CD-ROM. Le recuerdo, por lo tanto, que en el ***CD-ROM directorio html\editor\tutor*** encontrará un completo tutorial sobre HTML 3.2. Si ya sabe algo de HTML y desea profundizar en el aprendizaje de este lenguaje, le recomiendo lea el documento "Referencia HTML 3.2", también incluido en el ***CD-ROM directorio html\html32***. Este documento incluye la DTD de HTML 3.2. Todos los lenguajes de marcas utilizan una DTD en el cual se definen todos sus elementos con las etiquetas y atributos que han de utilizar, por lo cual debe acostumbrarse a manejar y saber interpretar este tipo de documentos ya que son una de las bases de la tecnología XML. Uno de los objetivos de este capítulo es que comience a tomar contacto con las DTD y sepa reconocer las marchas en ellos utilizados para más

adelante aprender a crear sus propios DTD. En el apartado "Cómo interpretar una DTD" obtendremos las nociones básicas para comenzar.

ATRIBUTOS DE LOS ELEMENTOS HTML

Los elementos HTML pueden utilizar una serie de parámetros adicionales para establecer o modificar algunas características del formato que han de aplicar al contenido; estos parámetros se conocen con el nombre de atributos y van siempre en la etiqueta de inicio tras el nombre del elemento. Los elementos vacíos también pueden utilizar atributos. Una característica de HTML 3.2 es que cada elemento tiene definidos sus atributos particulares, aunque existen atributos que pueden ser utilizados por diversos elementos como en el caso ALIGN (alineación). Con HTML 4.0, como veremos en el próximo capítulo, se incorporan los denominados atributos comunes que pueden ser utilizados por prácticamente cualquier elemento.

Los valores asignados a los atributos en HTML se recomienda que vayan entre comillas, aunque no es obligatorio su uso; no obstante, debe recordar que *en los lenguajes basados en XML el valor de los atributos siempre ha de ir entre comillas*.

Ejemplos etiquetas con atributos:

```
<H1 ALIGN="CENTER">Título principal</H1>
<TABLE ALIGN="LEFT" WIDTH="90%" BORDER ="2"> ... </TABLE>
<IMG SRC="IMGS\FOTO.JPG" ALT="IMAGEN" ALIGN="MIDDLE">
```

ENTIDADES EN HTML

La marca utilizada por HTML para indicar que hacemos referencia a una entidad es **&** (*ampersand*) y termina en con ; (punto y coma). Se puede insertar una entidad en una página utilizando su nombre o el código que se le ha asignado. En los documentos HTML, por lo general, se utilizan entidades para poder incorporar caracteres especiales utilizados como marcas o caracteres con código ASCII superior a 127. Para valores superiores se utilizan entidades definidas en algún documento, como puede ser, por ejemplo, el ISO Latin-1. Las entidades de uso más frecuente son, por lo tanto, los referidos a:

- Caracteres especiales propios de marcas como **&** (entidad), **<** (inicio de etiqueta) y **>** (cierre de etiqueta).

- Caracteres para la ISO Latin-1, entre los cuales se incluyen símbolos como vocales acentuadas o específicos de una determinada lengua como la letra ñ (eñe) propia de la lengua española.

Si queremos, por ejemplo, mostrar el texto "La etiqueta <HTML> está destinada a indicar el inicio de un documento en este lenguaje", para poder mostrar las marcas < y > de la palabra HTML, hemos de utilizar las referencias < (menor que) y > (mayor que) para que no sea tomada como una etiqueta por el intérprete. Además, la a acentuada de la palabra "está" debe utilizar una entidad definida en el ISO de Latin-1 como aacute y, por ello, hemos de utilizar la referencia &aacute, tal como se muestra a continuación:

```
<P> La etiqueta &lt;HTML&gt; est&aacute; destinada a indicar el inicio de un
documento en este lenguaje </P>
```

Hemos de tener en cuenta que, aunque los caracteres de Latin-1 generalmente son interpretados correctamente por las últimas versiones de navegadores sin utilizar entidades, puede ser que navegadores antiguos muestren datos extraños al intentar interpretar este tipo de caracteres. En el siguiente apartado aprenderá más sobre entidades y cómo se definen.

CÓMO INTERPRETAR UNA DTD

HTML es un lenguaje cuyas etiquetas están ya predefinidas en la DTD correspondiente; también se dice que es una aplicación **SGML** (**Standard Generalized Markup Language *o Lenguaje de Marcado General Estándar***), ya que éste es el lenguaje base de todos los lenguajes de marcas utilizados en Internet y que, además, es el utilizado para la creación de las Definiciones de Tipo de Documento.

De una forma genérica, podemos decir que las DTD sirven para definir los elementos que utilizará un lenguaje, sus etiquetas y atributos. Las principales notaciones, instrucciones o comandos SGML, utilizadas en las DTD son:

- **<!ELEMENT...>** Sirve para definir el nombre de un elemento, uso de las etiquetas y contenido.

- **<!ATTLIST...>** Sirve para definir la lista de atributos utilizados por un determinado elemento.

- **<!ENTITY %...>** Sirve para definir entidades, las entidades pueden ser utilizadas por elementos y atributos

Mostramos a continuación unos ejemplos de cómo están definidos algunos elementos y sus atributos en la DTD de HTML 3.2 :

- Elemento párrafo P, etiqueta <P>

```
<!ELEMENT P - O (%text)*>
<!ATTLIST P
   align (left|center|right) #IMPLIED >
```

• Elemento de ruptura de línea BR, etiqueta
 de contenido vacío (EMPTY)

```
<!ELEMENT BR - O EMPTY >
<!ATTLIST BR
   clear (left|all|right|none) none
>
```

• Elemento para selección de fuente FONT, etiquetas <FONT>...</FONT>

```
<!ELEMENT FONT - - (%text)* >
<!ATTLIST FONT
   size CDATA  #IMPLIED
   color CDATA #IMPLIED
>
```

ELEMENTOS, ATRIBUTOS Y ENTIDADES

Podemos decir que son los elementos de uso más frecuente en las DTD y aquéllos que debería comenzar a distinguir. La forma de definirlos es la siguiente:

<!ELEMENT...>

Sirve para declarar un elemento, es decir, establecer cómo se han de usar sus etiquetas y el tipo de contenido que pueden utilizar.

Sintaxis:

<!ELEMENT NOMBRE ? ? (CONTENIDO)>

Parámetros utilizados

• NOMBRE: nombre del elemento y sus etiquetas.

• **? ?**: segundo y tercer parámetro sirven para indicar cómo se han de usar las etiquetas. He utilizado el símbolo de interrogación ("?") en las posiciones en las cuales se indica si ese elemento ha de utilizar etiqueta de inicio y/o de finalización obligatoriamente, o bien, su uso es optativo. La posición de la primera interrogación hace referencia a la etiqueta de inicio; la segunda, a la etiqueta de cierre. La interrogación, por lo tanto, en la práctica aparecerá sustituida por un guión ("-") en aquellos casos en los que el uso de la etiqueta sea obligatorio y por una O, en aquellos casos en los cuales el uso de la etiqueta sea optativo.

• (CONTENIDO): siempre entre paréntesis establece qué elementos pueden ser utilizados como contenido. En caso de que el contenido sea de tipo texto, encontraremos generalmente la ***entidad*** (%text).

Ejemplos:

A continuación se muestra cómo interpretar algunos de los elementos definidos en las recomendaciones de HTML 3.2

- <!ELEMENT P - O (%text)*>

Elemento P etiqueta de inicio (-), etiqueta final optativa (O) y contenido texto (%text).

- <!ELEMENT FONT - - (%text)* >

Elemento FONT etiqueta de inicio (-), etiqueta final (-) y contenido texto.

- <!ELEMENT HR - O EMPTY>

Elemento HR etiqueta de inicio (-), sin etiqueta final (O) y sin contenido.

- <!ELEMENT BR - O EMPTY>

Elemento BR etiqueta de inicio (-), sin etiqueta final (O) y sin contenido.

- <!ELEMENT (%heading) - - (%text;)*>

Elementos H1, H2, H3, H4, H5 y H6 etiqueta de inicio (-), etiqueta de cierre (-) y contenido texto.

<!ATTLIST...>

Sirve para declarar la lista de los atributos (parámetros) que puede utilizar un determinado elemento en su etiqueta.

Sintaxis:

<!ATTLIST ELEMENTO nombre valor #TIPO [nombre valor #TIPO]...>

El parámetro #TIPO permite indicar si existen valores predeterminados y, de ser así, cómo se utilizarán.

Ejemplos

- Declaración de los atributos del **elemento FONT**:

```
<!ATTLIST FONT
   size    CDATA   #IMPLIED
```

```
   color   CDATA   #IMPLIED
>
```

Esta declaración ATTLIST establece que el elemento FONT puede utilizar dos atributos: *size* y *color*, y que sus valores han de proporcionarse como una cadena de caracteres (CDATA, Character DATA o datos de tipo carácter). Además, indica que, en caso de no asignarse ningún valor al atributo, se utilizará el valor predeterminado (#IMPLIED). Los valores predeterminados pueden ser asignados por el intérprete del navegador, o bien, en la propia declaración del atributo entre paréntesis. En un caso práctico la etiqueta FONT, por lo tanto, podría tener el siguiente aspecto: <FONT COLOR ="#00FFFF" SIZE="5">.

• Declaración de atributos del **elemento P**

```
<!ATTLIST P
   align (left|center|right) #IMPLIED
>
```

Esta declaración establece que el elemento P puede utilizar el atributo ALIGN cuyos valores preestablecidos son *left, center, right* y, en caso de no proporcionarse ningún valor, se utilizará el valor predeterminado (*left*), es decir, el primero que figura en la lista. Ejemplo de uso: <P ALIGN="CENTER">Párrafo centrado...

• Declaración de atributos de los **elementos de título Hx** (definidos como endidad %heading):

```
<!ATTLIST ( %heading )
   align (left|center|right) #IMPLIED
>
```

Como podemos observar, los elementos H1, H2, etc., también utilizan el atributo ALIGN de forma similar al elemento P. Ejemplo: <H1 ALIGN="RIGHT">Título principal</H1>.

• Declaración de atributos del **elemento HR**

```
<!ATTLIST HR
   align   (left|right|center)     #IMPLIED
   noshade (noshade)       #IMPLIED
   size    %Pixels         #IMPLIED
   width   %Length         #IMPLIED
>
```

Según esto el elemento HR, línea horizontal, puede utilizar cuatro atributos: *align, noshade, size* y *width*. Ejemplo:

```
<HR ALIGN="CENTER" SIZE="200" HEIGHT="10" NOSHADE>
```

Como puede comprobar, los atributos en la DTD están declarados en minúsculas pero, por lo general y en la práctica, encontrará que en los documentos HTML aparecen escritos en mayúsculas. Ésta es otra muestra de la permisividad existente para este tipo de documentos a la hora de su interpretación.

XML, por el contrario, es sensible a mayúsculas y minúsculas por lo cual los elementos y atributos han de escribirse tal y como se han definido.

<!ENTITY >

Sirve para declarar entidades. Una entidad puede ser desde un simple carácter a todo un archivo externo y son expandidas por el intérprete. Por el momento diremos que una entidad permite definir una serie de valores (una lista de elementos o atributos, por ejemplo) que pueden ser utilizados por un determinado elemento o en la lista de sus atributos. Los nombres de las entidades de elementos o atributos siempre aparecen precedidos del carácter tanto por ciento ("%"). Ejemplos: %heading, %text.

Sintaxis:

<!ENTITY [%] nombre valor>

Ejemplos:

- Declaración de la **entidad %heading**,

```
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
```

Esa instrucción declara la entidad %heading a la cual se le asignan los nombres de los seis posibles elementos utilizados para títulos en los documentos HTML. La barra vertical que separa cada uno de los elementos corresponde al símbolo de OR exclusivo con lo cual se establece que cualquiera de los nombres proporcionados en la lista puede ser utilizado.

- Declaración de **caracteres especiales &**, **<** y **>**

```
<!ENTITY amp CDATA "&#38;" -- ampersand -->
<!ENTITY lt CDATA "&#60;"  -- less than -->
<!ENTITY gt CDATA "&#62;"  -- greater than -->
```

Declaraciones de entidades para los caracteres propios de la lengua española en ISO Latin-1:

```
<!ENTITY iexcl CDATA "&#161;" -- inverted exclamation mark -->
<!ENTITY iquest CDATA "&#191;" -- inverted question mark -->
<!ENTITY Aacute CDATA "&#193;" -- capital A, acute accent -->
<!ENTITY Eacute CDATA "&#201;" -- capital E, acute accent -->
```

```
<!ENTITY Iacute CDATA "&#205;" -- capital I, acute accent -->
<!ENTITY Oacute CDATA "&#211;" -- capital O, acute accent -->
<!ENTITY Uacute CDATA "&#218;" -- capital U, acute accent -->
<!ENTITY aacute CDATA "&#225;" -- small a, acute accent -->
<!ENTITY eacute CDATA "&#233;" -- small e, acute accent -->
<!ENTITY iacute CDATA "&#237;" -- small i, acute accent -->
<!ENTITY oacute CDATA "&#243;" -- small o, acute accent -->
<!ENTITY uacute CDATA "&#250;" -- small u, acute accent -->
<!ENTITY Ntilde CDATA "&#209;" -- capital N, tilde -->
<!ENTITY ntilde CDATA "&#241;" -- small n, tilde -->
```

En el documento de "Especificaciones de HTML 3.2" encontrará también la tabla con los códigos de los caracteres imprimibles de Latin-1.

Uso de entidades para declarar elementos y sus atributos

Las entidades, como hemos dicho, pueden ser utilizadas, por lo tanto, también para declarar:

- un grupo de elementos:

```
<!ELEMENT ( %heading ) - - (%text;)*>
```

- sus atributos correspondientes:

```
<!ATTLIST ( %heading)
    align (left|center|right) #IMPLIED
>
```

Tras la lectura de este apartado, usted ya puede comprender cómo se definen elementos, atributos y entidades en las DTD; por ello, le recomendaría abrir el documento "Especificación de Referencia para el HTML 3.2" del CD-ROM en el directorio ***html\html32\sp*** (en español) o ***html\html32\en*** (en inglés) y practicara con su lectura e interpretación. Más adelante aprenderá a crear sus propias DTD.

FORMATO DE UN DOCUMENTO CON HTML PASO A PASO

Tras esta introducción al lenguaje HTML y las DTD pasaremos a la parte práctica que consistirá en dar formato a nuestro artículo visto en el capítulo anterior. Para efectuar el marcado del documento, tendremos en cuenta dos cosas:

1.º la estructura del documento.

2.º las etiquetas que nos ofrece HTML 3.2 para efectuar el formato.

1. Estructura del documento

Antes de comenzar con la tarea de marcado del documento, recordaremos cuál era su estructura, ya vista en el capítulo anterior.

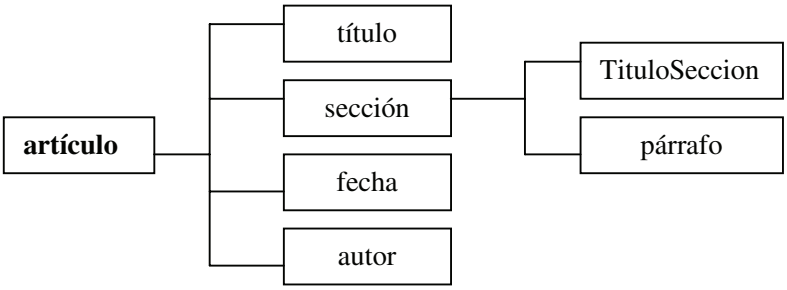

Figura 2 - 2. Representación gráfica de la estructura del documento

2. Selección de los elementos de html

Los diversos elementos existentes en la estructura del documento se pueden diferenciar por medio de elementos existentes en el lenguaje HTML y utilizar sus etiquetas para aplicar los correspondientes formatos al documento.

A) ELEMENTOS DE NIVEL BLOQUE

Para diferenciar cada uno de los elementos que aparecen en la estructura del documento, utilizaremos etiquetas HTML de **nivel de bloque** que permitirán diferenciar claramente cada uno de ellos de forma clara. Entre estos elementos tenemos los de títulos o cabeceras (H1, H2) y los delimitadores de párrafos (P). De forma esquemática lo podríamos representar de la siguiente forma:

- **Título**: etiquetas <H1> </H1>
- **Secciones** - TituloSeccion: etiquetas <H2> </H2>
  - párrafo: etiqueta <P>
- **Fecha:** párrafo, etiqueta <P>
- **Autor**: párrafo, etiqueta <P>

B) ELEMENTOS DE NIVEL TEXTO

Las etiquetas de nivel bloque de HTML aplican un formato predeterminado utilizando estilos predefinidos sobre su contenido. En ocasiones es necesario modificar determinadas partes de ese bloque para que aparezcan de forma distinta o resaltadas. HTML utiliza para ello los denominados elementos de nivel de texto (o *inline*).

Los elementos de nivel texto serán los que nos permitan poner en negrita y/o itálica determinadas partes del texto, cambiar el color, tamaño y tipo de letra, etc.

Los **elementos de nivel texto** que utilizaremos en nuestro documento son:

- <I> </I> para texto en itálica, inclinado.
- <B> </B> para texto en negrita (*bold*).
- <FONT> </FONT> para modificar color y tipo de la fuente.

Modificación de los estilos predeterminados

En HTML 3.2 existe la posibilidad de modificar los estilos predeterminados de un determinado bloque mediante el uso del elemento FONT, aunque, como vamos a ver, esta posibilidad presenta ciertas limitaciones. Recordando la definición de los atributos para este elemento en su DTD es la siguiente:

```
<!ATTLIST FONT
   size    CDATA #IMPLIED
   color   CDATA #IMPLIED
>
```

Es decir, según las recomendaciones de la W3C, solamente sería posible cambiar el tamaño y color pero no la fuente utilizada. Para establecer el color, se ha de asignar al atributo COLOR, tras el símbolo "#", un valor en hexadecimal correspondiente a la notación RGB (Red, Green, Blue); el rango de valores va de 0 a 255, es decir, dos dígitos hexadecimales con valores entre 00 y FF para cada uno de los colores:

FF0000 sería el rojo puro (R, *red*)

00FF00 sería el verde (G, *green*)

0000FF sería el azul (B, *blue*)

<H1><FONT COLOR="#FF0000">Lenguajes de marcas</FONT></H1> nos mostrará el título del documento con el color rojo puro.

<H2><FONT COLOR="#0000FF">SGML </FONT></H2> nos permitirá ir estableciendo el color azul para los títulos de las diversas secciones...

El principal problema nos lo encontramos a la hora de cambiar de fuente de letra ya que los títulos utilizan la fuente Comic Sans MS; la fecha y autor, la fuente Arial, y no existe ningún otro elemento HTML que permita seleccionar un determinado tipo de letra. Ante esta limitación se optó por añadir un nuevo atributo al elemento FONT que, aunque, como ya hemos visto, no está contemplado en la DTD del lenguaje, es implementado por la mayoría de los navegadores; este atributo es FACE. Así pues, nuestra nueva instrucción es la siguiente:

```
<H1><FONT COLOR="#FF0000" FACE="Comic Sans MS"> Lenguajes de
marcas <FONT></H1>
```

La cuestión de establecer el tamaño por medio del atributo SIZE presenta también ciertas limitaciones ya que solamente se pueden dar valores en un rango -7 y 7, siendo su valor por defecto 5.

3. Edición del documento y aplicación de formatos

A continuación y con todo lo visto hasta ahora podemos pasar a la práctica dando formato a nuestro artículo. Si desea hacerlo así, puede abrir el documento de ejemplo situado en el ***CD-ROM directorio Ejemplos\Cap02***, y que se encuentra en el archivo "**artículo.txt**". Para su edición puede utilizar el Bloc de Notas de Windows o cualquier otro editor. La práctica consiste en aplicar las etiquetas antes citadas para obtener el modelo visto en el Capítulo 1 o lo más parecido posible. Después guárdelo con el nombre de **articulo.htm**, o cualquier otro nombre, pero con la extensión **.htm** o **.html** para que sea reconocido como documento HTML por el explorador y así poder abrirlo automáticamente al efectuar un doble clic sobre él. Si utiliza el Bloc de Notas, procure seleccionar en *Tipo de archivo* la opción *todos los archivos *.** y escribir la extensión .htm para que no lo guarde con la extensión .txt (extensión por defecto). Si utiliza otro editor de textos, seleccione la opción *Guardar como...* → *Sólo texto* o alguna similar e indique además del nombre la extensión .htm.

4. Visualización y actualización del documento

El siguiente paso es abrir el documento HTML con el explorador que utilice para Internet y observar los resultados. Si quiere efectuar modificaciones y el navegador es Internet Explorer, puede hacerlas directamente desde la opción de menú "Ver\Código Fuente"; esto abrirá automáticamente el "Bloc de Notas"; tras modificar el documento, cierre el editor aceptando "Guardar los cambios" y pulse en el botón de "Actualizar" del explorador para que la página se recargue.

Si lo prefiere, puede abrir el archivo ***Ejemplos\Cap02\articulo.htm*** con el explorador de Internet y observar el código fuente; es el siguiente (se han resaltado en negrita las etiquetas del lenguaje HTML utilizadas para dar formato al documento).

artículo.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>Articulo HTML 3.2</TITLE>
</HEAD>
<BODY>
<H1 ALIGN="center" ><FONT COLOR="#FF0000" FACE="Comic Sans
MS">Lenguajes de marcas</FONT></H1>
<H2><FONT COLOR="#0000FF" FACE="Comic Sans
MS"><U>SGML</U></FONT></H2>
<P><B>Standard Generalized Markup Language</B> o <I><B>Lenguaje General
Estándar de Marcado</B></I> es el estándar para la creación y definición de otros
lenguajes de marcado de documentos. SGML es utilizado para la creación de los
denominados documentos de validación <B>DTD</B> ( <B>Document Type
Definition</B>  o <I><B>Definición de Tipo de Documento</B></I>) en los cuales se
definen los elementos, sintaxis y normas de un determinado lenguaje. SGML es, por
lo tanto, un metalenguaje utilizado para la creación de otros lenguajes.
<H2><FONT COLOR="#0000FF" FACE="Comic Sans
MS"><U>HTML</U></FONT></H2>
<P><B>HiperText Markup Language</B> o <I><B>Lenguaje de Marcado de
Hipertextos</B></I> es el estándar utilizado para la aplicación de formatos y
publicación de documentos en Internet. En HTML no hay separación entre contenido
(datos) y presentación (formato); el formato se aplica directamente sobre el contenido.
HTML es una aplicación SGML, una DTD, en el cual sus marcas (tags o etiquetas)
están ya definidas.
<H2><FONT COLOR="#0000FF" FACE="Comic Sans
MS"><U>XML</U></FONT></H2>
<P><B>Extensible Markup Language</B> o <I><B>Lenguaje de Marcado
Extensible</B></I>. XML se basa en SGML y es, por así decirlo, una versión reducida
de él. Al igual que SGML, es un lenguaje orientado a la definición de la estructura de
los documentos pero no a su representación (aplicación de formatos). XML es
también un metalenguaje. El lenguaje XML no tiene etiquetas predefinidas.
<H2><FONT COLOR="#0000FF" FACE="Comic Sans
MS"><U>XHTML</U></FONT></H2>
<P><B>Extensive HiperText Markup Language</B> o <I><B>Lenguaje de Marcado
de Hipertextos Extensible</B></I>. Es un lenguaje basado en XML; podemos decir
que es el lenguaje HTML pero aplicando la normativa utilizada por XML. XHTML está
orientado a la creación de documentos bien estructurados y aplicación de formatos.
<P><FONT FACE="Arial">Sábado, 4 de enero de 2003</FONT>
<P><FONT FACE="Arial"><I>Alonso Rodríguez Zamora</I></FONT>
</BODY>
</HTML>
```

DOCUMENTO HTML RESULTANTE

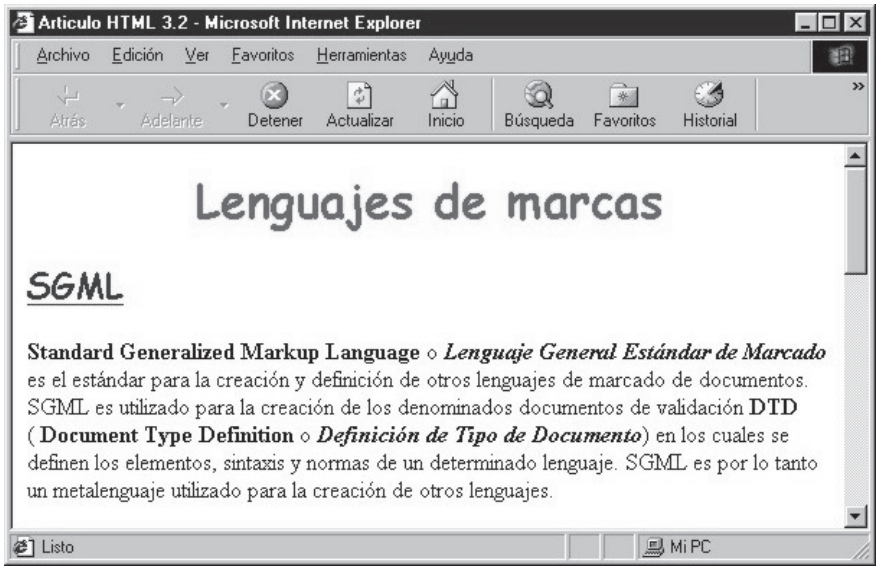

Figura 2 - 3. Visualización del documento HTML

Como puede observar en la figura 2 - 3, el resultado es muy parecido al indicado en el modelo; no obstante, no es del todo idéntico; observe principalmente la ausencia de sangría en los comienzos de párrafo ya que, como hemos dicho, HTML carece de elementos de formato para aplicar sangrías o tabulaciones.

LIMITACIONES DE HTML 3.2

1. En la aplicación de estilos

Como podemos comprobar, el uso del elemento FONT para definir y aplicar estilos no ofrece muchas posibilidades; de hecho, es un elemento desaprobado (*deprecated*) en HTML recomendándose la utilización de las **CSS** (**Cascading Style Sheets**, ***Hojas de Estilo en Cascada***). Más adelante dedicaremos un capítulo a este tema.

2. En sangrías y espacios en blanco.

En HTML no existe etiqueta o atributo que posibilite establecer sangrías o tabulaciones para los párrafos. Como mucho, para conseguir sangrar un párrafo, podemos recurrir al truco de utilizar elementos de tipo lista (OL, UL, etc.) e incluir como contenido el texto que queremos sangrar, o bien, recurrir a las listas de definiciones (DL). Así, por

ejemplo, para conseguir una sangría francesa de un párrafo, podríamos utilizar la siguiente instrucción:

```
<UL>
Este texto se mostrará como si utilizara sangría francesa<BR>
pues es contenido de una lista.
</UL>
```

Encontramos también problemas cuando queremos dejar varios espacios en blanco de separación entre palabras de un párrafo pues HTML convierte todos los espacios existentes en uno solo. Las posibles soluciones son:

- El elemento PRE (párrafo preformateado), pero éste además aplica unos estilos propios, tipo de letra, tamaño, etc.

- La entidad definida en Latin-1 como **nbsp** (*no-break space*), la cual permite insertar un espacio en blanco extra dentro de un párrafo.

El uso de la entidad **nbsp** también permite efectuar sangrías de primer párrafo colocando varios   al comienzo de él. Ejemplo:

```
    Texto con sangría de primera línea<BR>
de cuatro caracteres.
```

USO DE ENTIDADES

Como hemos podido comprobar, el simple uso de etiquetas del lenguaje HTML no siempre permiten obtener el resultado deseado; por ello, se ha de recurrir al uso de entidades.

En nuestro documento utilizaremos la entidad ** ** para conseguir las sangrías que aparecen en el documento modelo colocando varios   seguidos en el inicio de los párrafos que deban ir sangrados. Las nuevas líneas presentarían el siguiente aspecto:

<P>   <B>Standard Generalized Markup Language</B>...

También es conveniente el uso de entidades de Latin-1 para las vocales acentuadas ya que visualizadores antiguos o que no efectúen automáticamente la interpretación de estos caracteres mostrarían símbolos extraños al visualizar el documento. Para las vocales minúsculas acentuadas las entidades son **á é í ó** y **ú**.

El código fuente de la nueva versión de nuestro documento se encuentra en el ***CD-ROM directorio Ejemplos\cap02\articulo2.htm***; las modificaciones efectuadas son las

correspondientes a sangrías de primer párrafo y caracteres acentuados; a continuación, en el contenido de dicho archivo se remarcan en negrita las entidades utilizadas.

articulo2.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>Articulo HTML 3.2</TITLE>
<META NAME="GENERATOR" CONTENT="CastWeb v1.1">
</HEAD>
<BODY>
<H1 ALIGN="center"><FONT COLOR="#FF0000" FACE="Comic Sans
MS">Lenguajes de marcas</FONT></H1>
<H2><FONT COLOR="#0000FF" FACE="Comic Sans
MS"><U>SGML</U></FONT></H2>
<P>        <B>Standard
Generalized Markup Language</B> o <I><B>Lenguaje General Est&aacute;ndar de
Marcado</B></I> es el est&aacute;ndar para la creaci&oacute;n y definici&oacute;n
de otros lenguajes de marcado de documentos. SGML es utilizado para la
creaci&oacute;n de los denominados documentos de validaci&oacute;n
<B>DTD</B> ( <B> Document Type Definition</B>  o <I><B>Definici&oacute;n de
Tipo de Documento</B></I>) en los cuales se definen los elementos, sintaxis y
normas de un determinado lenguaje. SGML es, por lo tanto, un metalenguaje utilizado
para la creaci&oacute;n de otros lenguajes.
<H2><FONT COLOR="#0000FF" FACE="Comic Sans MS"> <U>HTML</U>
</FONT></H2>
<P>        <B>HiperText
Markup Language</B> o <I><B>Lenguaje de Marcado de Hipertextos</B></I> es el
est&aacute;ndar utilizado para la aplicaci&oacute;n de formatos y
publicaci&oacute;n de documentos en Internet. En HTML no hay separaci&oacute;n
entre contenido (datos) y presentaci&oacute;n (formato); el formato se aplica
directamente sobre el contenido. HTML es una aplicaci&oacute;n SGML, un DTD, en
el cual sus marcas (tags o etiquetas) est&aacute;n ya definidas.
<H2><FONT COLOR="#0000FF" FACE="Comic Sans
MS"><U>XML</U></FONT></H2>
<P>         <B>Extensible
Markup Language</B> o <I><B>Lenguaje de Marcado Extensible</B></I>. XML se
basa en SGML y es por as&iacute; decirlo una versi&oacute;n reducida de
&eacute;l. Al igual que SGML, es un lenguaje orientado a la definici&oacute;n de la
estructura de los documentos pero no a su representaci&oacute;n
(aplicaci&oacute;n de formatos). XML es tambi&eacute;n un metalenguaje. El
lenguaje XML no tiene etiquetas predefinidas.
<H2><FONT COLOR="#0000FF" FACE="Comic Sans
MS"><U>XHTML</U></FONT></H2>
<P>         <B>Extensive
HiperText Markup Language</B> o <I><B>Lenguaje de Marcado de Hipertextos
Extensible</B></I>. Es un lenguaje basado en XML, podemos decir que es el
```

lenguaje HTML pero aplicando la normativa utilizada por XML. XHTML est**á** orientado a la creaci**ó**n documentos bien estructurados y aplicaci**ó**n de formatos.
<P><FONT FACE="Arial">S**á**bado, 4 de enero de 2003</FONT>
<P><FONT FACE="Arial"><I>Alonso Rodr**í**guez Zamora</I></FONT>
</BODY>
</HTML>

HERRAMIENTAS Y UTILIDADES HTML

1. Editores

Como ya hemos dicho y ha podido comprobar un simple editor de texto, es suficiente para la creación de documentos HTML o páginas Web. No obstante, existen herramientas que nos pueden ayudar en la generación más rápida de los documentos al permitirnos aplicar de forma automática las etiquetas y atributos correspondientes. Estas herramientas van desde los más sofisticados entornos de desarrollo visuales a sencillos editores HTML. Si lo que desea es aprender el lenguaje HTML, mi consejo es que comience por el uso de estos últimos.

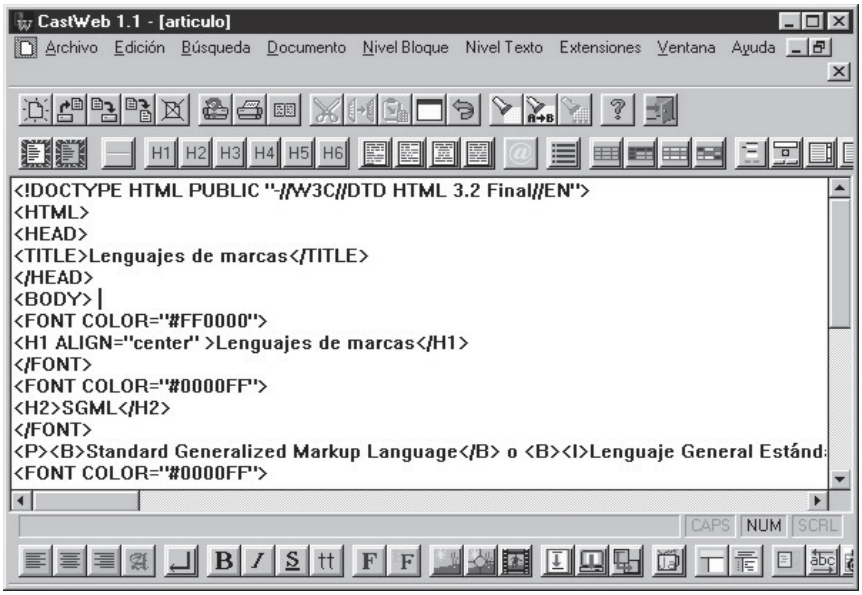

Figura 2 - 4. El editor de HTML 3.2 CastWeb

En el ***CD-ROM directorio "html\editor"*** incluyo un programa que hice hace ya algún tiempo, **CastWeb**, un editor HTML implementado para la versión HTML 3.2, el estándar por entonces (figura 2 - 4). CastWeb no es un entorno visual de desarrollo Web, es simplemente un editor de HTML y está especialmente diseñado para enseñar a aprender HTML; le ayudará a insertar las etiquetas necesarias y le indicará los atributos que puede utilizar. CastWeb va acompañado de un archivo de ayuda en formato .hlp con el cual puede aprender fácilmente su uso; además, incluye un tutor de HTML que le puede servir también para introducirse en la creación de páginas Web de una forma sencilla.

2. Tutoriales y documentación

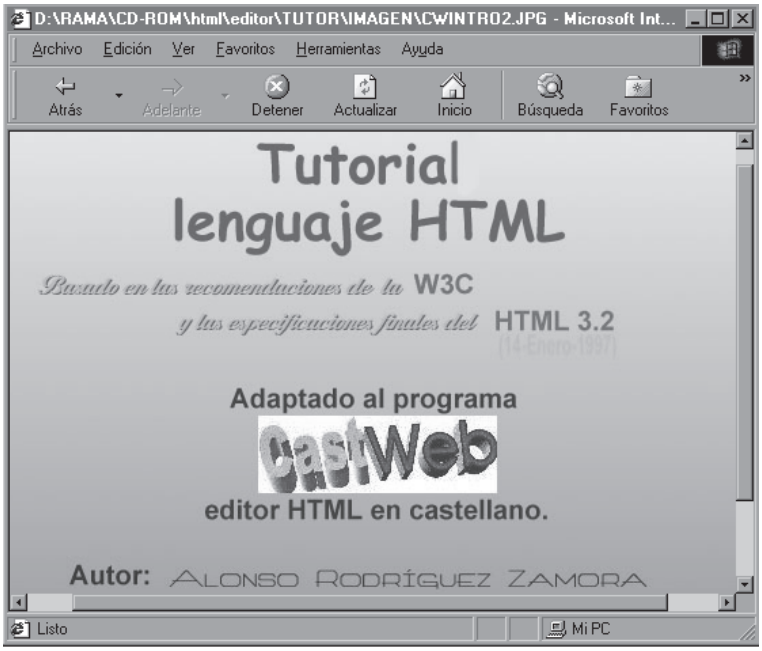

Figura 2 - 5. Tutorial de HTML 3.2

En el ***CD-ROM directorio "HTML\editor\tutor\indice.htm"*** se incluye un tutorial de HTML 3.2 que le puede servir perfectamente de complemento para este capítulo; además, al estar en formato HTML, la visualización del código fuente también puede servirle de ayuda. Con este tutor podrá aprender HTML y el uso del editor de HTML CastWeb.

CONCLUSIÓN

HTML es un lenguaje de marcas, una aplicación SGML, principalmente diseñado para el formateo y composición de los documentos que se han de publicar en Internet. Basta un simple vistazo a la DTD en la cual se encuentra definido este lenguaje para comprobar que la mayoría de sus marcas o etiquetas están relacionadas con la ***aplicación de formatos*** preestablecidos a bloques de texto o caracteres individuales. Existen otras etiquetas destinadas a la ***maquetación o composición*** para permitir la colocación de una forma más ordenada de sus contenidos; se recurre para ello a la utilización de diversos tipos de listas <OL>, <UL>, etc., y principalmente al uso de tablas <TABLE>.

Debemos tener en cuenta que el contenido de un documento no solamente es texto, sino que existen también otros elementos que pueden ser incorporados como su contenido: entre ellos están las imágenes, sonidos, vídeos o animaciones (elementos multimedia). Para superar sus propias limitaciones, HTML ha tenido que recurrir a la utilización de elementos y tecnologías externas como pueden ser el uso de lenguajes de *scripts*, hojas de estilo, *applets*, controles ActiveX, etc.

No debemos olvidar nunca la finalidad con la cual ha sido diseñado este lenguaje: aplicar formatos y composición de documentos. HTML y XML no tienen ningún punto en común ni es posible efectuar, por lo tanto, comparaciones entre ambos ya que, como veremos, *XML ha sido diseñado para trabajar con las estructuras de los documentos*, no con su representación visual.

HTML y XML pueden trabajar de forma conjunta y complementaria; de hecho, en la práctica suelen hacerlo como iremos viendo a lo largo de este libro. En realidad, cuando vamos un poco más allá de la publicación de un determinado documento en Internet e intentamos crear un sitio Web, vemos que se hace necesario el uso y combinación de diversas tecnologías como pueden ser las hojas de estilo, los lenguajes de *scripts* o, incluso, la utilización de auténticos lenguajes de programación como Java.

COMPLEMENTOS

Al final de los capítulos encontrará por lo general el apartado COMPLEMENTOS; en él se hará una referencia sobre aplicaciones, utilidades, documentación, tutoriales, etc., que le permitirán profundizar en los temas tratados.

En el CD-ROM

html \editor\CastWeb.exe Editor de HTML 3.2.

html\editor\tutor\indice.htm Tutorial de HTML 3.2.

html\html32\en\ HTML 3_2 Reference Specification.htm Referencia de HTML 3.2 y DTD, el documento original del W3C en inglés.

html\html32\sp\Referencia HTML 3_2.htm Referencia de HTML 3.2 y DTD en castellano. Traducción de Gustavo Zamboni.

html\html3.2\wilbur\wilbur31.hlp Referencia HTML 3.2 en inglés en formato ayuda de Windows de Web Design Group (WDG).

visores\ie55 Microsoft Internet Explorer 5.5 español.

visores\ie6 Microsoft Internet Explorer 6 español.

En Internet

World Wide Web Consortium (W3C) http://www.w3c.org/

W3C HTML Home Page: http://www.w3.org/MarkUp/

W3C Recomendación HTML 3.2 http://www.w3.org/TR/REC-html32/

W3C Recomendaciones y documentos técnicos http://www.w3.org /TR/

Referencia HTML 3.2 en castellano: http://dns.uncor.edu/info/html/rec-sp.htm

Web Design Group (WDG) http://www.htmlhelp.com/

CAPÍTULO 3

PUBLICACIÓN CON HTML DINÁMICO

INTRODUCCIÓN

Antes de entrar en el presente tema, efectuaré una aclaración: con el uso de HTML Dinámico se pretende principalmente poder generar los contenidos de los documentos de forma dinámica y que el usuario pueda interactuar con dichos documentos. Microsoft y Netscape utilizan tecnologías distintas para la generación de documentos dinámicos. El HTML Dinámico utilizado por Microsoft se basa en **DOM** (**Document Object Model** o ***Modelo de Objetos de Documento***), mientras que el utilizado por Netscape se basa en los tipos dinámicos de tecnología TrueDoc. Sin entrar en discusión sobre cuál de los métodos es el mejor, nosotros utilizaremos el modelo propuesto por Microsoft por considerar que es el más adecuado al contenido de la obra, ya que DOM es también la base para la programación de documentos XML y, además, por el momento, Internet Explorer, a partir de su versión 5, es el navegador que mejor soporte ofrece al lenguaje XML. Así pues, cuando utilice el término HTML Dinámico, me estaré refiriendo al utilizado por Microsoft. El uso de hojas de estilo y lenguajes de *script*, aunque las implementaciones puedan variar, sí son comunes a ambos navegadores.

Los **objetivos** del presente capítulo son los siguientes:

1. Analizar las principales diferencias existentes entre el HTML estático, basado en las Recomendaciones HTML 3.2 visto en el capítulo anterior, y el HTML Dinámico, basado en las Recomendaciones HTML 4.0 y las novedades incorporadas por éstas a dicho lenguaje.

2. Proporcionar unos conocimientos básicos sobre las tecnologías utilizadas por DHTML tales como hojas de estilo, lenguajes de *script*, DOM y su uso práctico.

3. Mostrar que la publicación en Internet no se basa en la utilización de "un" determinado lenguaje, sino de diversas tecnologías que se complementan.

HTML DINÁMICO

DHTML (**Dynamic HTML** o ***HTML Dinámico***) está basado en el uso de las Recomendaciones de HTML 4.0 y en DOM, el cual originariamente se denominaba en realidad "Modelo de Objetos de DHTML". DOM surgió como un intento de superar las limitaciones propias del HTML estático, basado, como hemos visto, en las recomendaciones de HTML 3.2, y poder transformarlo en HTML Dinámico. Por medio de DOM los elementos de HTML, además de estar destinados a efectuar el formato del documento, se convierten en ***objetos con propiedades y métodos que además pueden reaccionar ante una serie de eventos***, sucesos tales como ser señalados por el cursor del ratón (OnMouseOver), pulsaciones efectuadas en el botón del ratón (OnMouseDown) o del teclado (OnKeyDown), etc., de esta forma ***los elementos HTML pasan a convertirse en entidades programables***. Para poder acceder a ellos y ser programados, todo elemento (objeto) ha de tener un nombre o identificador; por ello, en DHTML son de máxima importancia los atributos NAME e ID, incorporados en HTML 4.0 como ***atributos comunes*** para poder ser utilizados por casi todos sus elementos. La programación se efectúa por medio de lenguajes de *script*, tales como JavaScript y VBScript.

DOM se convirtió en estándar con una Recomendación del 1 de octubre de 1998 del W3C. En el ***CD-ROM directorio dom\DOMLevel-1***puede encontrar ese documento distribuido libremente por dicha organización. En el directorio ***dom\dom1-es*** se encuentra la traducción al español, por Juan R. Pozo, de ese documento.

REC-DOM-Level-1-19981001

Document Object Model (DOM) Level 1 Specification Version 1.0

W3C Recommendation 1 October, 1998

http://www.w3.org/TR/REC-DOM-Level-1

Los documentos basados en HTML Dinámico se caracterizan, como ya hemos dicho, porque los elementos que los componen pueden ser programados y, de esta forma, cambiar su comportamiento o, incluso, pueden crearse documentos completos nuevos en tiempo de ejecución (o visualización), es decir, de una forma dinámica, ya que, cuando se abre un documento DHTML con un navegador con soporte DOM, el intérprete se encarga de construir tantos objetos como etiquetas HTML existan en él.

La tecnología de HTML Dinámico se basa en:

- El lenguaje de marcas HTML 4.0 o HTML 4.01
- Las Hojas de estilo (CSS)

- Los lenguajes de *script* o guiones (lenguajes interpretados), JavaScript y VBScript, principalmente.

- Incorporación de programas externos de lenguajes compilados, *applets* de Java y controles ActiveX, principalmente.

- *Data binding*, generación de páginas dinámicamente (manipulación, ordenación, filtrado) con contenidos de bases de datos.

- Amplio soporte multimedia.

- Uso de navegadores de la cuarta generación: entre ellos tenemos Microsoft Internet Explorer, Netscape Navigator en sus versiones 4 o superior.

Como puede comprobar, el HTML Dinámico es bastante complejo y no se refiere solamente a la inclusión de animaciones en páginas Web, sino que se basa principalmente en la utilización de una serie de tecnologías que se complementan y en la programación. La publicación de documentos DHTML implica una serie de conocimientos, además del lenguaje HTML 4, tales como la definición de estilos mediante CSS, programación con lenguajes de *script* como mínimo y, a un nivel más avanzado, conocimiento de la interfaz DOM y algún lenguaje de programación compilado (Java, Visual Basic) si deseamos incluir soporte extra como Applets o componentes ActiveX.

En este capítulo nos centraremos en las novedades aportadas por las Recomendaciones HTML 4.0, el uso de los lenguajes de *script* y un breve estudio de los objetos DOM. El próximo capítulo estará dedicado por completo a las hojas de estilo en cascada (CSS), ya que su conocimiento resulta casi imprescindible para la publicación de documentos basados tanto en HTML como en XML.

NOVEDADES DE HTML 4.0

HTML 4.0 se convirtió en estándar por medio de una Recomendación del W3C en diciembre de 1997.

REC-html40-971218

HTML 4.0 Specification W3C Recommendation 18-Dec-1997

http://www.w3.org/TR/REC-html40/

En el ***CD-ROM directorio html\Html40\Html40.pdf*** tiene ese documento en formato PDF; para su lectura, necesitará el programa de Adobe Acrobat Reader; por, si no dispone de él, y por ser de libre distribución, también ha sido incluido en el ***CD-ROM directorio utilidades\Acrobat.*** Además, en el ***CD-ROM directorio html\Html40hlp*** se incluye una versión del WDG en formato HELP de Windows.

Existe una recomendación posterior para HTML 4.01 que no incorpora novedades al lenguaje, sino que es una simple remodelación de él. En el ***CD-ROM directorio html\Html40\Html401*** tiene ese documento en formato HTML; y en el ***directorio html\Html40\Html401-es*** una traducción al español de Juan R. Pozo.

HTML 4.01 Specification W3C Recommendation 24 December 1999

http://www.w3.org/TR/html401

La principal característica de HTML 4.0 es que es un lenguaje claramente orientado a la creación de documentos basados en el uso de tecnología DHTML. Así, por ejemplo, para la aplicación de formatos a los documentos, se recomienda el uso de hojas de estilo CSS y todos sus elementos pueden ser programados mediante lenguajes de *script*. Las nuevas etiquetas y atributos están diseñados principalmente para dar soporte a esas dos tecnologías. Otra característica es que ya no existen elementos abiertos, es decir, todo elemento con contenido ha de utilizar etiqueta de inicio y final: <P>... párrafo... </P>, <LI> Elemento de lista </LI>, etc.; estos elementos, como vimos en el capítulo anterior, podían omitir la etiqueta de cierre según las recomendaciones existentes para HTML 3.2. Se utilizará, por lo tanto, una sola etiqueta con aquellos elementos que sean declarados como vacíos (EMPTY) en la DTD.

Con HTML 4.0 también se pretende poner un poco más de orden en la publicación de documentos en Internet y en la utilización lo más fiel posible del estándar propuesto por parte de los desarrolladores de intérpretes. Se va imponiendo la idea de "validación de documentos" (concepto propio de las tecnologías basadas en XML), es decir, la comprobación de que un documento basado en un DTD, especificado en la línea de prólogo mediante <!DOCTYPE...>, siga fielmente la sintaxis y uso de los elementos en él establecidos.

TIPOS DE DOCUMENTOS HTML 4.0

Según las recomendaciones de la W3C, todo documento HTML debe comenzar con una declaración de tipo de documento, línea de prólogo, en la cual se indica la versión de HTML 4.0 utilizada. Existen tres tipos de DTD con una declaración DOCTYPE distinta para cada una de ellas:

HTML 4.0 Strict

En HTML 4.0 Estricto prevalece la idea de estructura del documento sobre la presentación o formato. Los elementos y atributos designados como desaprobados (*deprecated*) en otras versiones o Recomendaciones de HTML no están permitidos. Es la DTD utilizada por HTML 4.0 por defecto.

La línea de declaración de tipo de documento (DOCTYPE) para aquéllos que utilicen esta DTD es la siguiente:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">

HTML 4.0 Transitional

HTML 4.0 Transitional incluye todos los elementos y atributos de HTML 4.0 Strict, pero permite utilizar los elementos desaprobados. La declaración de este tipo de documentos es:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">

HTML 4.0 Frameset

HTML 4.0 Frameset es una variante de HTML 4.0 Transitional para documentos que usen *frames*. El elemento BODY es reemplazado por un elemento FRAMESET en un documento Frameset. La declaración de este tipo de documentos es:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN" "http://www.w3.org/TR/REC-html40/frameset.dtd">

Forma genérica de indicar el uso de HTML 4.0 y DTD

Se puede establecer como tipo de documento HTML 4.0 e indicar como URI (Uniform Resource Identifier) la última versión existente para una determinada DTD o un archivo de entidades con las siguientes direcciones:

- "http://www.w3.org/TR/REC-html40/strict.dtd" (DTD Estricta)
- "http://www.w3.org/TR/REC-html40/loose.dtd" (DTD Transicional)
- "http://www.w3.org/TR/REC-html40/frameset.dtd" (DTD de *Frames*)
- "http://www.w3.org/TR/REC-html40/HTMLlat1.ent" (Entidades Latin-1)
- "http://www.w3.org/TR/REC-html40/HTMLspecial.ent" (Entidades Especiales)
- "http://www.w3.org/TR/REC-html40/HTMLsymbol.ent" (Entidades Símbolos)

Ejemplos:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN" "http://www.w3.org/TR/REC-html40/frameset.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN" "http://www.w3.org/TR/REC-html40/HTMLlat1.ent"

VALIDACIÓN DE DOCUMENTOS HTML

Teniendo en cuenta cada una de las anteriores DTD, es posible efectuar validaciones de los documentos basados en HTML 4.0. El W3C posee una página en Internet conocida como **HTML Validation Service** (http://validator.w3.org/) desde la cual se pueden efectuar validaciones de documentos HTML y XHTML. También existen utilidades de validación en Internet de libre distribución, entre ellas "CSE HTML Validator" (http://www.htmlvalidator.com) incluida en el ***CD-ROM directorio html\validador\cselite***.

NUEVAS ETIQUETAS EN HTML 4.0

HTML 4.0 sigue utilizando elementos existentes en la versión de HTML 3.2 y añade otros nuevos; algunos de ellos ya habían sido implementados por los navegadores aunque no eran estándares. A continuación se citan esos nuevos elementos.

Elementos utilizados en *frames*

FRAMESET etiquetas <FRAMESET>..</FRAMESET>

FRAME etiqueta <FRAME> sin contenido (EMPTY)

NOFRAMES etiquetas <NOFRAMES>...</NOFRAMES>

Estructura de los frames:

```
<FRAMESET>
        <FRAME....>
        [<FRAME....> ]
</FRAMESET>
<NOFRAMES>
  ... contenido visualizadores que no soporten frames ....
</NOFRAMES>
```

Elementos de nivel texto (*inline*)

ABBR - Abreviación

```
<ABBR TITLE="eXtensive Markup Languaje">XML</ABBR>
```

ACRONYM - Acrónimo

```
<ACRONYM TITLE="Organización del Tratado del Atlántico Norte">OTAN
</ACRONYM>
```

BDO BiDirectional Override; establece la dirección de escritura del texto

```
<BDO dir="ltr">texto</BDO>
```

BUTTON - Botón genérico; permite incluir texto e imagen en él

```
<BUTTON id="ayuda" ACCESSKEY=A OnClick="ayudar()" > Ayuda <IMG
SRC="help.gif"> </BUTTON>
```

IFRAME - *frames* en línea (Frame Inline)

```
<IFRAME src="prueba.html" width="300" height="100">
<!--Contenido alternativo para browsers sin soporte IFRAME  -->
<H2>Titulo página HTML</H2>
<P>... parrafada... </P>
</IFRAME>
```

INS - Texto insertedo

Q - Anotación corta (Short quotation)

S - Texto tachado (Strike-through)

SPAN - Contenedor genérico *inline*, especialmente diseñado para aplicación de estilos. Este elemento será analizado ampliamente más adelante.

Elementos de nivel bloque

DIV - Contenedor genérico, especialmente diseñado para aplicación de estilos. También será analizado ampliamente más adelante. Debemos tener en cuenta que DIV ya existía en HTML 3.0; para efectuar división de texto, no obstante, sus capacidades eran muy limitadas ya que solamente permitía modificar la alineación mediante su único atributo ALIGN; por ello, era un elemento apenas utilizado.

NOSCRIPT - Contenido alternativo a un *script*.

OBJECT - Especialmente indicado para la inclusión de controles ActiveX.

Elementos de tablas

COLGROUP - Grupo de Columnas

COL - Columna

THEAD - Cabecera de la Tabla (Table HEAD)

TFOOT - Pie de la Tabla (Table FOOT)

TBODY - Cuerpo de la Tabla (Table BODY)

Elementos de formularios

FIELDSET - Grupo de controles de un Formulario

LEGEND - Título del Fieldset

LABEL - Etiqueta de un campo del Formulario

OPTGROUP - Grupo de Opciones

ATRIBUTOS COMUNES PARA ETIQUETAS HTML 4.0

Una de las modificaciones más interesantes que encontramos con respecto a la versión HTML 3.2 es en la revisión de los atributos utilizados por las etiquetas; algunas de las existentes desaparecen o pasan a ser consideradas como desaprobadas (*deprecated*) según el término utilizado en las Recomendaciones W3C y surgen un grupo de atributos denominados de uso común o **atributos comunes**, es decir, que puede ser utilizado por cualquier elemento de HTML. Los atributos comunes son una de las principales características de la nueva forma de entender HTML; para una mejor comprensión de su funcionamiento diferenciaremos los siguientes grupos:

Atributos para diferenciar los objetos

Atributos comunes que permiten diferenciar los diversos elementos (objetos) existentes en el documento asignándoles un nombre (atributo NAME) o un identificador (atributo ID). Estos atributos, por lo general, se utilizan solamente cuando ese objeto va a ser programado mediante un lenguaje de *script*.

NAME

Este atributo sirve para asignar un nombre a un objeto HTML:

```
<P name="NOP">Objeto párrafo con nombre NOP</P>
```

ID

Este atributo sirve para asignar un identificador único a un elemento (objeto) HTML :

```
<P id="PID">Objeto párrafo con identificador PID</P>
```

Atributos para definir el estilo del elemento

Estos atributos forman el grupo de los denominados "atributos comunes principales (Core)" según las recomendaciones de la W3C y están claramente enfocados a ser utilizadas con hojas de estilo. HTML 4.0 recomienda la utilización de estilos CSS; de hecho, el elemento FONT con el cual se seleccionaba en la versión anterior las fuentes y color pasa a ser un elemento desaprobado.

STYLE

Este atributo permite aplicar directamente el estilo indicado al definirse dentro del propio elemento. Ejemplo:

<P style="font-family: Verdana"> Texto con fuente de letra Verdana </P>

CLASS

Atributo utilizado con estilos definidos en bloques <STYLE> </SYTLE> u hojas de estilo externas y cuyo selector es una clase, es decir, al definirse su nombre, va precedido de punto ("."). Ejemplo:

```
.marino {
  color: navy;
}
```

Desde la página HTML se aplicaría el estilo con una instrucción similar a:

<P class="marino">Párrafo con texto en azul marino.</P>

ID

Atributo utilizado con estilos definidos en bloques <STYLE></SYTLE> u hojas de estilo externas y cuyo selector es un identificador, es decir, al definirse su nombre, va precedido del carácter almohadilla ("#"). Ejemplo:

```
#marino {
  color: navy;
}
```

Desde la página HTML se aplicaría el estilo con una instrucción similar a:

<P id="marino">Párrafo con texto en azul marino.</P>

El atributo **id** puede crear confusión ya que puede referirse al identificador de un objeto (elemento HTML) o al identificador de un estilo; por ello, es más recomendable definir los estilos como clases. El atributo **id** no es soportado por Netscape Navigator hasta las versiones 6.x.

Atributos de internacionalización

LANG

El atributo LANG especifica el lenguaje utilizado por un determinado elemento. Los posibles valores de este atributo se encuentran en el documento RFC 1766; algunos de sus posibles valores son, por ejemplo:

en para inglés (English)
en-US para inglés americano
ja para Japonés

fr para Francés
sp para Español

Ejemplo:

<P lang="en">This paragraph is in English.</P>

DIR

El atributo DIR especifica la dirección en que se ha de escribir el texto. Sus posibles valores son **dir**="ltr" (*left-to-right*), de izquierda a derecha, que es el valor por defecto y **dir** ="rtl" (*right-to-left*), de derecha a izquierda.

Otros atributos comunes

TITLE

El atributo TITLE proporciona el título a un elemento; este título puede ser mostrado como un "tooltip" en los *browsers* visuales. Puede ser de gran utilidad con los elementos A, LINK, IMG y OBJECT, proporcionando información sobre el recurso enlazado. También es valioso su uso con abreviaciones y acrónimos, es decir, los elementos ABBR y ACRONYM:

<ABBR title="HiperText Markup Language">HTML</ABBR>

<ACRONYM title="Organización del Tratado del Atlántico Norte"> OTAN </ACRONYM>

EVENTOS COMUNES DE LOS OBJETOS HTML 4.0

Como hemos dicho al comienzo del capítulo, los elementos HTML pueden reaccionar a determinados eventos o sucesos. De forma similar a lo que sucede con los atributos comunes, todos los elementos de HTML 4.0 pueden reaccionar a una serie de sucesos denominados **eventos comunes**. Naturalmente existen elementos que reaccionan ante determinados eventos particulares como es, por ejemplo, el caso de BODY con los eventos onLoad (al efectuarse la carga) y onUnload (al abandonar la página). Los eventos comunes son los siguientes:

onClick, se produce cuando el botón del ratón es pulsado sobre un elemento.

onDblClick, cuando el botón del ratón es pulsado dos veces sobre un elemento.

onMouseDown, cuando el botón del ratón es pulsado sobre un elemento.

onMouseUp, cuando el botón del ratón es soltado estando sobre un elemento.

onMouseOver, cuando el puntero del ratón pasa sobre un elemento.

onMouseMove, cuando el puntero del ratón se mueve estando sobre un elemento.

onMouseOut, cuando el puntero del ratón se aleja de un elemento.

onKeyPress, cuando una tecla es pulsada y soltada sobre un elemento.

onKeyDown, cuando una tecla es pulsada sobre un elemento.

onKeyUp, cuando una tecla es soltada sobre un elemento.

La respuesta o modo de comportarse de estos objetos ante un determinado evento se efectúa mediante la programación de las denominadas funciones para tratamiento de eventos, o funciones gestoras de eventos, que utilizan la siguiente sintaxis genérica:

onEvento="respuesta( )"

Ejemplo de función gestora para el evento OnClick; se ejecuta al pulsar con el botón del ratón sobre el elemento:

onClick="alert('Me has pulsado')"

Ejemplo de botón que muestra un mensaje al ser pulsado:

<BUTTON onClick="alert('Me has pulsado')" > Púlsame </BUTTON>

Éste es un ejemplo muy sencillo que utiliza una función ya predefinida de JavaScript, alert('mensaje'); este lenguaje es el utilizado por defecto para la programación de funciones gestoras de eventos; como es natural, la función solamente se ejecutará si el navegador utilizado incorpora el intérprete JavaScript.

Cuando se utilizan funciones propias, definidas por el programador, éstas se encuentran entre de las etiquetas contenedoras de *scripts* <SCRIPT> y </SCRIPT> en el bloque de cabecera (HEAD) o en archivos independientes que han de ser enlazados.

A continuación veremos cómo publicar documentos con HTML 4.0 utilizando la tecnología de las hojas de estilo y, posteriormente, unos sencillos ejemplos de cómo se definen y utilizan *scripts* para gestión de eventos de elementos HTML con VBScript y JavaScript.

PUBLICACIÓN DE DOCUMENTOS CON HTML 4.0

Como hemos visto anteriormente, una de las principales características de HTML 4.0 es la incorporación de nuevas etiquetas y atributos comunes especialmente destinados a proporcionar soporte a las hojas de estilo en cascada CSS; este tema será estudiado a fondo en el próximo capítulo; en el presente apartado se muestra cómo utilizar el atributo STYLE para incorporar nuevos estilos a los elementos HTML o modificar los formatos establecidos por defecto. Para ello escribiremos una nueva versión de nuestro artículo utilizando la DTD de HTML 4.0 Stric, es decir, eliminando todos los elementos desaprobados, tales como el atributo ALIGN o el elemento FONT que, como vimos, mostraba grandes deficiencias para aplicar o definir estilos.

CÓDIGO FUENTE

El documento formateado mediante HTML 4.0 se encuentra en el ***CD-ROM ejemplos\cap03\articulo.htm*** y su contenido es el siguiente:

articulo.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD>
<TITLE>Articulo HTML 4.0</TITLE>
<META http-equiv=Content-Type content="text/html; charset=iso-8859-1">
</HEAD>
<BODY>
<H1 style="font-family:Comic Sans MS; font-size:16pt; color:#FF0000; text-
align:center">Lenguajes de marcas</H1>
<H2 style="font-family:Comic Sans MS; font-size:14pt; font-weight:normal;
color:#0000FF"><U>SGML</U></H2>
<P style="font-size:11pt; text-indent:4em"><B>Standard Generalized Markup
Language</B> o <I><B>Lenguaje General Estándar de Marcado</B></I> es el
estándar para la creación y definición de otros lenguajes de marcado de documentos.
SGML es utilizado para la creación de los denominados documentos de validación
<B>DTD</B> (<B>Document Type Definition</B>  o <I><B>Definición de Tipo de
Documento</B></I>) en los cuales se definen los elementos, sintaxis y normas de un
determinado lenguaje. SGML es, por lo tanto, un metalenguaje utilizado para la
creación de otros lenguajes.</P>
<H2 style="font-family:Comic Sans MS; font-size:14pt; font-weight:normal;
color:#0000FF"><U>HTML</U></H2>
<P style="font-size:11pt; text-indent:4em"><B>HiperText Markup Language</B> o
<I><B>Lenguaje de Marcado de Hipertextos</B></I> es el estándar utilizado para la
aplicación de formatos y publicación de documentos en Internet. En HTML no hay
separación entre contenido (datos) y presentación (formato); el formato se aplica
```

```
directamente sobre el contenido. HTML es una aplicación SGML, un DTD, en el cual
sus marcas (tags o etiquetas) están ya definidas.</P>
<H2 style="font-family:Comic Sans MS; font-size:14pt; font-weight:normal;
color:#0000FF"><U>XML</U> </H2>
<P style="font-size:11pt; text-indent:4em"><B>Extensible Markup Language</B>
o <I><B>Lenguaje de Marcado Extensible</B></I>. XML se basa en SGML y es, por
así decirlo, una versión reducida de él. Al igual que SGML, es un lenguaje orientado a
la definición de la estructura de los documentos pero no a su representación
(aplicación de formatos). XML es también un metalenguaje. El lenguaje XML no tiene
etiquetas predefinidas.</P>
<H2 style="font-family:Comic Sans MS; font-size:14pt; font-weight:normal;
color:#0000FF"><U>XHTML</U></H2>
<P styule="font-size:11pt; text-indent:4em"><B>Extensive HiperText Markup
Language</B> o <I><B>Lenguaje de Marcado de Hipertextos Extensible</B></I>. Es
un lenguaje basado en XML, podemos decir que es el lenguaje HTML pero aplicando
la normativa utilizada por XML. XHTML está orientado a la creación de documentos
bien estructurados y aplicación de formatos.</P>
<P style="font-family:Arial; font-size:10pt">Sábado, 4 de enero de 2003</P>
<P style="font-family:Arial; font-size:10pt; text-indent:2em"><I>Alonso Rodríguez
Zamora</I></P>
</BODY>
</HTML>
```

En el listado aparecen resaltados los elementos característicos de HTML 4.0; podemos destacar los siguientes puntos:

- Declaración del tipo de documento en el cual está basada la DTD utilizada por este documento.

- Establecer que el contenido del documento utilizará el juego de caracteres definido en Latin-1 (charset=iso-8859-1)

- Definición de estilos, por medio del atributo **style**, para la aplicación de formatos. Debemos tener en cuenta que HTML 4.0 Estricto utiliza las hojas de estilo CSS de forma genérica para la definición y aplicación de los estilos a los documentos. El uso del elemento FONT se hace innecesario; de hecho, era uno de los elementos desestimados (*deprecated*) y se permite su uso en documentos de HTML 4.0 Transicional solamente.

- Definición de sangrías para la primera línea de párrafo por medio de la propiedad de hojas de estilo **text-indent**, con lo cual se hace innecesario el uso de la entidad   utilizada en el capítulo anterior para conseguir este tipo de sangrías.

- Todos los elementos con contenido aparecen cerrados y utilizan etiqueta de inicio y finalización.

• Por convención los nombres de los elementos en las etiquetas van en mayúsculas y sus atributos en minúscula. Son de esta forma como están definidos en las DTD también.

DOCUMENTO RESULTANTE

Puede comprobar que, por medio del atributo **style**, ya hemos obtenido unos resultados muy similares al documento propuesto en el modelo. Incluso hemos conseguido aplicar el sangrado de la primera línea de los párrafos explicativos. Puede comparar el código HTML con el del ejemplo del capítulo anterior para ver las diferencias existentes. De momento no se preocupe si no entiende bien las definiciones de los estilos ya que será tratado en el siguiente capítulo en profundidad.

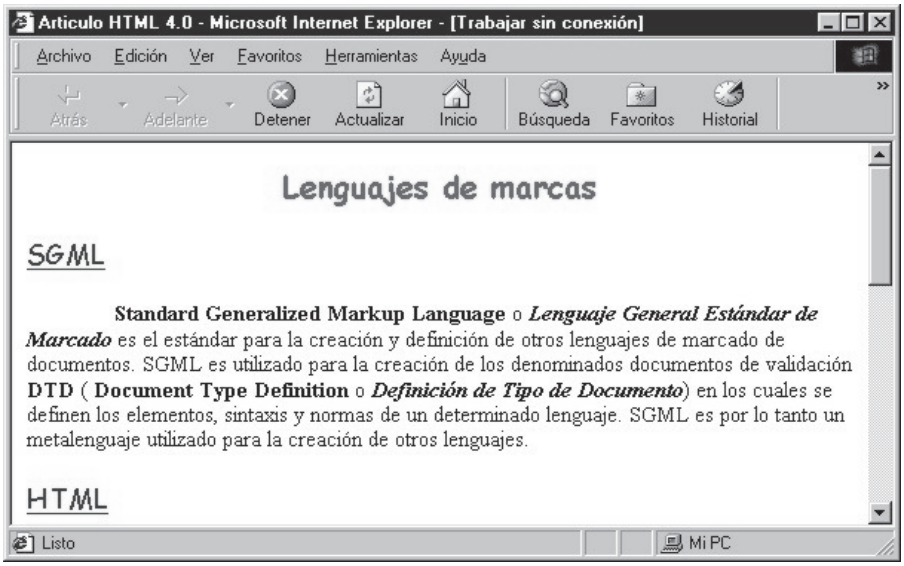

Figura 3 - 1. Resultado obtenido con el atributo style

LENGUAJES DE *SCRIPT*

Son utilizados para programar el comportamiento de los objetos HTML Dinámico al reaccionar ante un determinado evento. Los lenguajes de *script* son todos interpretados; por ello, solamente se ejecutarán las funciones utilizadas por el documento si el navegador incorpora el intérprete correspondiente al lenguaje utilizado. El lenguaje considerado como estándar para la programación de los objetos HTML es JavaScript; no obstante, pueden utilizarse otros; entre ellos destaca VBScript de Microsoft, que, naturalmente, es soportado por todos los navegadores de esta casa.

El código del *script* va siempre en formato de texto normal, texto sin formato y, como norma general, podemos decir que no se requiere de ninguna herramienta especial para la creación de *scripts*, aunque existen herramientas que pueden ayudarnos a su desarrollo y depuración.

Scripts en el propio documento

Los *scripts* pueden ir formando parte del documento HTML como contenido del elemento SCRIPT y, por lo general, en el bloque de cabecera (HEAD). Además, en la etiqueta se ha de indicar por medio del atributo LANGUAGE el lenguaje utilizado por el *script*. En el caso de JavaScript también es recomendable utilizar el número de versión. Ejemplos:

```
<SCRIPT LANGUAGE ="JavaScript1.3">
< ! –
    function saludar() {
        alert( "Bienvenido a mis páginas personales." );
    }
-->
</SCRIPT>

<SCRIPT LANGUAGE="VBScript">
<!--
    Sub saludar()
        MsgBox "Bienvenido a mis páginas personales."
    End Sub
-->
</SCRIPT>
```

Como prevención para los navegadores que no soporten *scripts*, se puede utilizar el elemento NOSCRIPT con el contenido alternativo a su uso.

```
<NOSCRIPT>
... contenido para navegadores que no incorporen el intérprete del script ...
</NOSCRIPT>
```

Scripts en documentos externos (enlazados)

Los *scripts*, sobre todo si son muy amplios o han de ser utilizados por diversos documentos, suelen ir en archivos independientes, que son enlazados desde la página HTML por medio del elemento SCRIPT y asignando a su atributo SRC (*source*) el nombre, o localización, del archivo con los *scripts*. Ejemplos:

```
<SCRIPT SRC="jscripts.js" LANGUAGE ="JavaScript"></SCRIPT>
```

<SCRIPT SRC="vbscripts.vbs" LANGUAGE ="VBScript"></SCRIPT>

Por convención, los archivos con *scripts* de JavaScript utilizan la extensión ".js" y los de VBScript, ".vbs". A continuación se verán unos sencillos ejemplos de cómo se utilizan estos lenguajes en los documentos de HTML Dinámico.

UTILIZACIÓN DE VBSCRIPT EN PÁGINAS HTML

Uso de funciones predefinidas

Al igual que ocurre con JavaScript, en VBScript existen funciones que están predefinidas en el lenguaje y, por lo tanto, pueden ser utilizadas directamente. La función alert() vista anteriormente es de JavaScript con VBScript; podríamos utilizar la función MsgBox() propia de este lenguaje. Ambas hacen más o menos lo mismo: muestran una ventana con el mensaje indicado. Ejemplos:

```
<BUTTON OnClick="MsgBox( 'Me has pulsado' )" > Púlsame </BUTTON>
<FORM>
<INPUT TYPE="BUTTON" VALUE="Púlsame" OnClick=' MsgBox "Pulsado botón."'>
</FORM>
```

Observe la diferencia también entre el elemento BUTTON, un nuevo elemento *inline* propio de HTML4.0, y el elemento INPUT TYPE="BUTTON" propio de HTML3.2 y utilizado en los formularios (FORM).

Definición de subrutinas y funciones propias

Como ya hemos dicho, se implementan en el bloque de cabecera (HEAD) del documento entre las etiquetas <SCRIPT> y </SCRIPT>

En VBScript existe diferencia entre subrutinas (Sub) y funciones (Function). Las primeras simplemente ejecutan determinadas acciones, las segundas pueden devolver un determinado valor como resultado. Ejemplo:

```
<HEAD>
<TITLE>Uso de VBScript</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!—
Sub saludar()
    MsgBox "Bienvenido a mis páginas personales."
End Sub
Sub informar()
    MsgBox "Botón pulsado."
```

```
End Sub
Sub Boton1_OnClick
    MsgBox "Botón de formulario pulsado."
End Sub
-->
</SCRIPT>
</HEAD>
```

Ejecución de los *scripts*

La ejecución se efectúa desde el bloque del cuerpo (BODY) del documento por medio de alguno de los objetos existentes en él en respuesta a un determinado evento. Ejemplo:

```
<BODY OnLoad="saludar()">
... resto del documento ...
<BUTTON OnClick="informar()" > Púlsame </BUTTON>
<FORM>
<INPUT TYPE="BUTTON" NAME="Boton1" VALUE="Púlsame">
</FORM>
</BODY>
```

Debemos observar que, para este último ejemplo, no existe definido un evento, OnEvento=, al cual responder; esto es debido a que se utiliza el atributo NAME="Boton1" y la respuesta para este elemento ya ha sido establecida al declarar la subrutina: Sub Boton1_OnClick. Ésta es una característica propia de VBScript al igual que la mostrada a continuación.

Scripts personalizados (SCRIPT... FOR... EVENT...)

Aunque no es frecuente, también es posible definir los *scripts* en el bloque de cuerpo de documento y establecer una relación FOR="nombre_elemento" y EVENT="onEvento". Ejemplo dentro de un formulario.

```
<FORM NAME="Form1">
<INPUT TYPE="Button" NAME="Boton1" VALUE="Púlsame">
<SCRIPT LANGUAGE="VBScript" FOR="Boton1" EVENT="onClick" >
    MsgBox "Boton Pulsado!"
</SCRIPT>
</FORM>
```

A continuación se muestra un sencillo ejemplo completo con las posibles formas de definir y ejecutar VBScripts en documentos HTML.

En el ***CD-ROM directorio Ejemplos\Cap03\VBScript.html*** encontrará el código fuente completo; es el siguiente:

VBScript.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD><TITLE> Ejemplos VBScript </TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub saludar()
    MsgBox "Bienvenido a mis páginas personales."
End Sub

Function pregunta()
    pregunta = InputBox( "¿Cómo te llamas?" )
End Function

Sub Button1_OnClick
    MsgBox "Botón pulsado."
End Sub
-->
</SCRIPT>
</HEAD>
<BODY OnLoad="saludar()">
<H3> Ejemplos VBScript  </H3>
<HR>
<BUTTON OnClick="MsgBox( 'Hola ' & pregunta )"> Pregunto... </BUTTON>
<HR>
<FORM>
<INPUT NAME="Button1" TYPE="BUTTON" VALUE="Púlsame">
</FORM>
<FORM NAME="Form2">
  <INPUT TYPE="Button" NAME="Boton2" VALUE="Púlsame">
  <SCRIPT LANGUAGE="VBScript" FOR="Boton2" EVENT="onClick" >
    MsgBox "Boton Pulsado!"
  </SCRIPT>
</FORM>
</BODY>
</HTML>
```

Utilización de JavaScript en páginas HTML

JavaScript es el lenguaje de *scripts* más ampliamente utilizado para el desarrollo de guiones en documentos DHTML. Entre sus principales características está el ser un lenguaje basado en objetos y, por lo tanto, puede ser un buen punto de partida para

aquéllos que deseen introducirse en el mundo de la programación orientada a objetos y, como su nombre indica, está inspirado en el lenguaje Java. JavaScript tiene ya definidos una serie de objetos con sus correspondientes propiedades y métodos; además, permite la creación de nuevos objetos.

Existe cierta semejanza entre algunos objetos definidos en JavaScript y los definidos en DOM de HTML; lo veremos en el próximo apartado, concretamente los denominados "objetos del navegador" y que son los siguientes: window, location, history y document. Según esto podemos observar que JavaScript es un lenguaje especialmente diseñado para ser utilizado en documentos Web; de hecho, estos objetos pueden representar la estructura de un documento HTML como podemos ver en el esquema que se muestra a continuación. Jerarquía de los "objetos del navegador" de JavaScript:

- **window [parent, frames, self, top]**
  - **history**
  - **location**
  - **document**
    - **anchor**
    - **links**
    - **form**
      - **button**
      - **checkbox**
      - **hidden**
      - **radio**
      - **reset**
      - **select**
      - **submit**
      - **text**
      - **textarea**

Por medio de estos objetos JavaScript, se pueden crear de forma dinámica, en tiempo de visualización, ventanas y documentos nuevos, y por medio del método write() del objeto documento, generar el contenido de dicho documento HTML.

PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

Por si no conoce este tipo de programación, a continuación se muestran los conceptos básicos de este tipo de programación. Un objeto es una entidad que posee unos determinados propiedades y métodos.

1. Las propiedades sirven para indicar el estado del objeto.

2. Los métodos representan las acciones que puede efectuar ese objeto; se utilizan también para modificar o acceder a las propiedades de los objetos.

Para referirnos a una propiedad o un método de un objeto, la sintaxis genérica es:

objeto.propiedad

objeto.método()

Para una mejor comprensión, a continuación se muestran las propiedades y métodos utilizados por el objeto document de JavaScript ya que, además, es el que más nos interesa por estar íntimamente relacionado con la publicación HTML.

EL OBJETO *document*

PROPIEDADES

alinkColor valor del atributo ALINK de <BODY>

Anchor representa un elemento ancla (A NAME=). ***Objeto***

anchors lista de objetos Anchor

Applet representa un elemento APPLET. ***Objeto***

applets lista de elementos Applet

Area área de un mapa de imagen, representa un elemento AREA. ***Objeto***

bgColor valor del atributo BGCOLOR de <BODY>

classes establece el estilo de un elemento, utilizado en <STYLE>

cookie representa una *cookie*, texto almacenado en cookie.txt

domain nombre del "dominio" del servidor

fgColor valor del atributo TEXT de <BODY>

Form representa un elemento FORM. ***Objeto***

forms lista de objetos Form

ids modifica el estilo de un elemento, utilizado en <STYLE>

Image representa un elemento IMG. ***Objeto***

images	lista de objetos Image
lastModified	valor con la fecha y hora de la última modificación
Layer	capa, representa un objeto LAYER (Netscape)
layers	lista de objetos Layer
linkColor	valor del atributo ALINK de <BODY>
Link	representa un objeto de enlace, ancla (A HREF=). ****Objeto***
links	lista de objetos Link
location	dirección URL del documento
referrer	URL del documento desde el cual se ha efectuado el hiperenlace
tags	establece el estilo de un elemento concreto, utilizado en <STYLE>
title	título del documento, elemento TITLE
URL	dirección URL del documento
vlinkColor	valor del atributo VLINK de <BODY>

***Objeto** esta notación indica que, aunque son utilizados como propiedades esos elementos, son también objetos, es decir, tienen sus propias propiedades y métodos. El uso de los objetos Anchor, Applet, Area, Form y Link como propiedades de document es para no utilizar sistemas jerárquicos diferentes, pero, en realidad, son objetos descendientes de *document*.

MÉTODOS

captureEvents()	retorna el evento capturado
clear()	borrar el documento
close()	cerrar el documento
getSelection()	retorna el texto seleccionado
handleEvents()	activa un determinado evento
open()	abre un documento

write() escribe en el documento

writeln() escribe una línea en el documento

En los ejemplos que siguen se muestran cómo utilizar algunos de estas propiedades y métodos.

HTML DINÁMICO CON JAVASCRIPT

El **objeto window** posee una serie de métodos para la generación de diversos tipos de ventanas, entre ellas las denominadas ventanas o cuadros de diálogo. Estos métodos son:

window.alert() para crear cuadros de mensaje.

window.confirm() cuadros de confirmación con las opciones OK y Cancel.

window.prompt() para entrada de datos.

Como muestra de aplicación basada en HTML Dinámico utilizando JavaScript, crearemos un interfaz desde el cual podremos abrir nuestro artículo o mostrar información correspondiente al entorno que estamos utilizando, con información sobre el navegador, documento, etc.

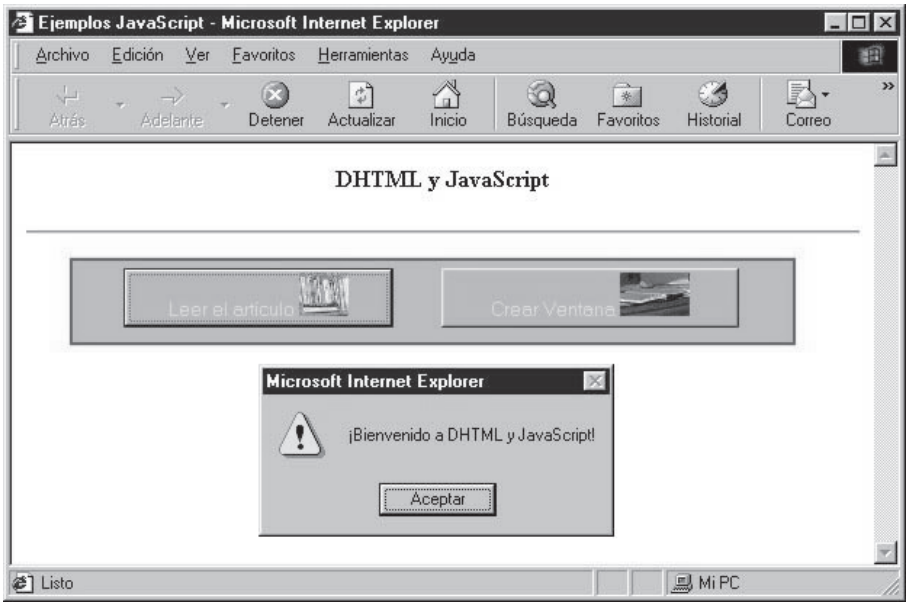

Figura 3 - 2. Interfaz basado en DHTML y JavaScript

La creación de ventanas nuevas con determinados documentos es una tarea fácil utilizando JavaScript y el método **window.open()**.

El código utilizado para la apertura de la ventana con nuestro artículo es el siguiente:

```
<SCRIPT LANGUAGE="JavaScript">
<!--
// función para abrir el documento html con el artículo en una ventana nueva
function abreVentana() {
    var ventana;

    ventana=open("articulo3a.htm","Articulo","toolbar=no,directories=no,menubar=no,s
    crollbars=yes");
}
-->
</SCRIPT>
```

window.open() Este método abre una ventana nueva con el documento que le indiquemos en su primer parámetro. El segundo parámetro sirve para establecer el título; el tercero es una cadena en la cual se indican diversos valores indicando las características propias de las ventanas estándar.

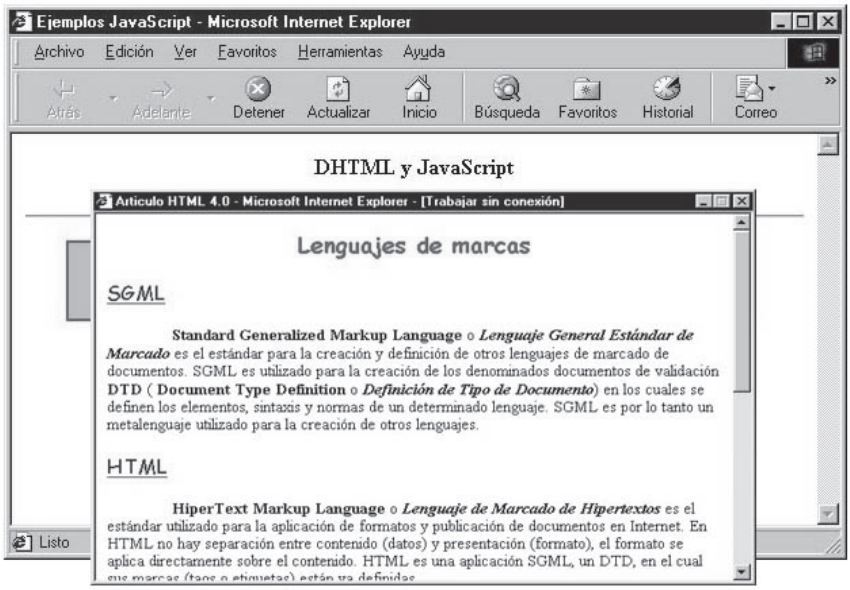

Figura 3 - 3. Creación de nuevas ventanas de forma dinámica

Cuando se pulsa el botón "Leer artículo", se genera de forma dinámica, utilizando código JavaScript, una nueva ventana cuyo contenido es el documento correspondiente a nuestro artículo, en concreto, el archivo "artículo.htm".

En el ***CD-ROM directorio Ejemplos\Cap03\interfaz.htm*** encontrará el código fuente para la visualización del documento en otra ventana generada dinámicamente. Es el siguiente:

interfaz.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD><TITLE> Ejemplos JavaScript </TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
// función para abrir el documento html con el artículo en una //ventana nueva
function abreVentana() {
var ventana;
ventana=open("articulo3a.htm","Articulo","toolbar=no,directories=no,menubar=no,
scrollbars=yes");
}
// función que crea un nuevo documento con información
function creaVentana() {
var ventana;
ventana=open("","Dinamica","toolbar=no,directories=no,menubar=no,scrollbars=yes");
ventana.document.write("<HTML><HEAD><TITLE>Ventana dinámica
</TITLE></HEAD>");
ventana.document.write("<BODY BGCOLOR='#FFFF00'
TEXT='#0000FF'><h1>Datos del documento padre</h1>");
ventana.document.write("Titulo del documento: <B>" +document.title+ "</B>.<BR>")
ventana.document.write("Última modificación del documento: <B>"
+document.lastModified+ "</B>.<BR>")
ventana.document.write("Color de fondo: <B>" +document.bgColor+ "</B>.<BR>")
ventana.document.write("Color de texto: <B>" +document.fgColor+ "</B>.<BR>")
ventana.document.write("Color de los enlaces: <B>" +document.linkColor+
"</B>.<BR>")
ventana.document.write("Color del enlace activo: <B>" +document.alinkColor+
"</B>.<BR>")
ventana.document.write("Color de los enlaces visitados: <B>" +document.vlinkColor+
"</B>.<BR>")
ventana.document.write("URL de la página: <B>" +document.location+ "</B>.<BR>")
ventana.document.write("<H3>Datos de este documento (Generado dinámicamente
con JavaScript)</H3>");
ventana.document.write("Titulo del documento: <B>" +ventana.document.title+
"</B>.<BR>")
ventana.document.write("Color de fondo: <B>" +ventana.document.bgColor+
"</B>.<BR>")
```

```
ventana.document.write("Color de texto: <B>" +ventana.document.fgColor+
"</B>.<BR>")
ventana.document.write("<H3>Información del navegador utilizado</H3>");
ventana.document.write("Nombre del navegador: <B>" +navigator.appName+
".</B><BR>")
ventana.document.write("Version: " +navigator.appVersion+ ".<BR>")
ventana.document.write("Nombre codificado: " +navigator.appCodeName+ ".<BR>")
ventana.document.write("Datos de la cabecera " +navigator.userAgent+ "." );
ventana.document.write("<BR></BODY></HTML>");
}
-->
</SCRIPT>
<STYLE TYPE="text/css">
.boton{
    color: yellow;
    background-color: orange;
    position: relative;
    left: 35;
    top: 5
}
.cajazul{
    background-color: cyan;
    width: 500;
    height: 60;
    border: 2px;
    border-style: solid;
    border-color: red;
    position: relative;
    left: 30;
    top: 10;
}
</STYLE>
</HEAD>
<BODY OnLoad='alert("¡Bienvenido a DHTML y JavaScript!");'>
<SCRIPT LANGUAGE="JavaScript">
<!-- ocultación a browsers sin soporte JavaScript
    document.write("<H3 ALIGN='center'> DHTML y JavaScript  </H3>");
final ocultación -->
</SCRIPT>
<NOSCRIPT>
Su navegador no da soporte a JavaScript
</NOSCRIPT>
<HR>
<DIV CLASS="cajazul">
<BUTTON CLASS="boton" OnClick="abreVentana();">
Leer el artículo <IMG SRC="leer2b.jpg" ALT="imagen leer" HEIGHT="30">
</BUTTON>
<SPAN STYLE="position:relative; left:30">
```

```
<BUTTON CLASS="boton" OnClick="creaVentana();"> Crear Ventana <IMG
SRC="leer.jpg" ALT="imagen leer" HEIGHT="30"> </BUTTON>
</SPAN>
</DIV>
</BODY>
</HTML>
```

Existe también la posibilidad de crear nuevas ventanas y generar su contenido, el documento, de forma dinámica. Como ejemplo de creación de documentos HTML de forma dinámica mediante JavaScript, incluimos una aplicación destinada a proporcionar información sobre el entorno que estamos utilizando y otros datos de interés. Este código se ejecutará al pulsar sobre el segundo botón de nuestro interfaz con el texto "Crear Ventana" y es el utilizado en la función creaVentana().

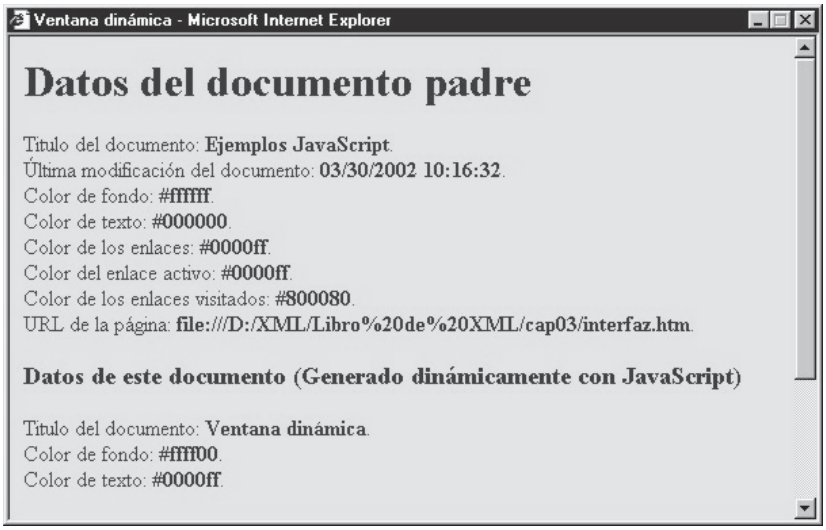

Figura 3 - 4. Creación de contenido y ventana de forma dinámica

Un botón abre una ventana con un documento existente y el otro abre una ventana y crea un documento nuevo con información sobre el documento padre, el documento activo y el navegador, imagen sobre estas líneas.

DOM HTML

Como hemos dicho anteriormente, cada elemento HTML, representado por un *tag* o etiqueta, de un navegador con soporte DOM se convierte en un objeto, el cual pasa a formar parte de la estructura del objeto documento que es el elemento base desde el cual se puede acceder a todos los demás elementos hijo.

Internet Explorer 4.0 ya incorporaba una jerarquía de objetos DOM para HTML. El modelo de objetos para HTML utilizado por este navegador es el representado en la siguiente figura:

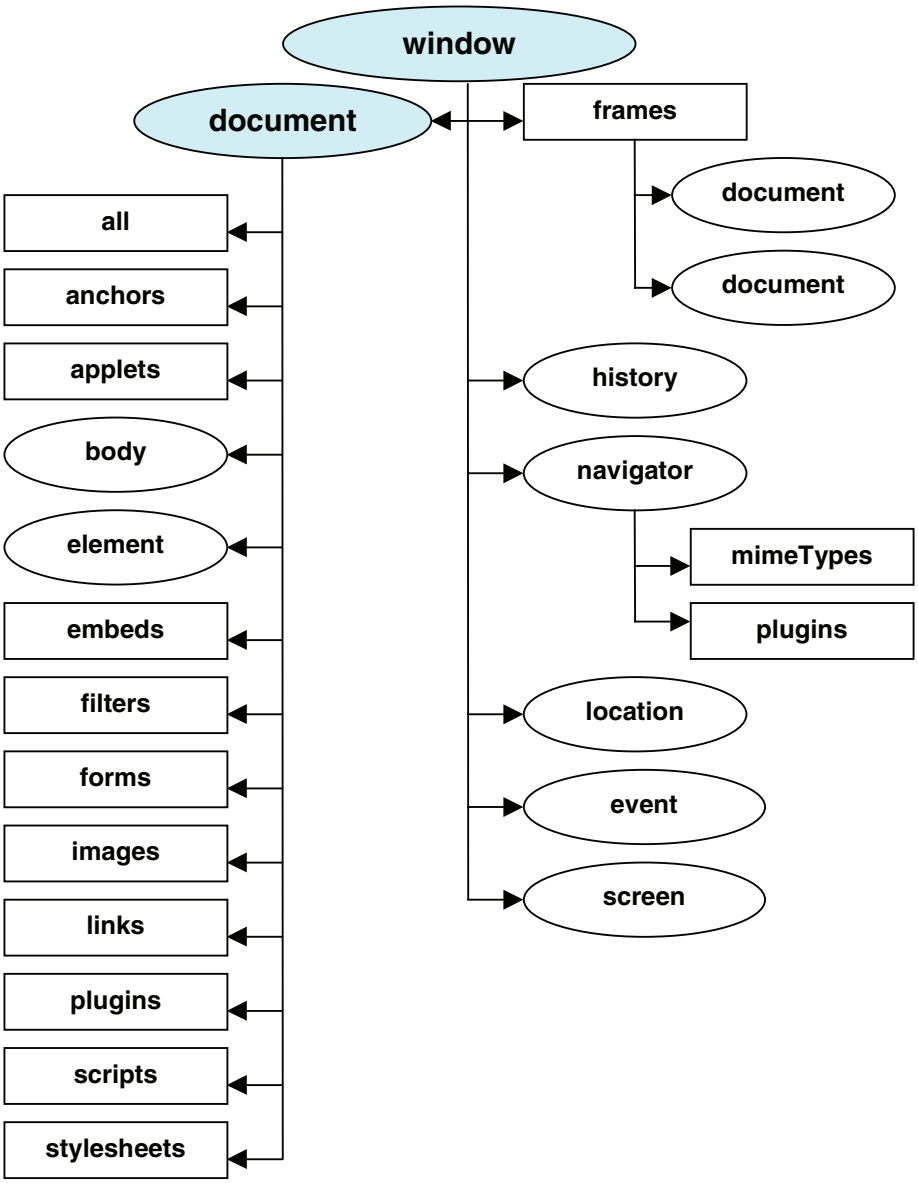

Figura 3 - 5. Esquema gráfico de DOM de objetos de HTML

Símbolos utilizados en el esquema DOM:

• **elipse**: objeto individual, con propiedades y métodos propios

• **rectángulo**: colección, lista de un determinado tipo de objeto

Por medio de el Modelo de Objetos de Documento (DOM) se puede crear una estructura, que puede ser representada en forma de árbol jerárquico, del documento. Cada elemento existente en el documento (cada etiqueta HTML) recibe el nombre de nodo en el árbol jerárquico.

El nodo raíz (objeto *document*) es el elemento principal de todo documento HTML y está representado por la etiqueta <HTML>. Los nodos que contienen otros elementos reciben el nombre de nodo padre (ramas de un árbol) y los elementos contenidos en otros nodos reciben el nombre de nodos hijo. Así, por ejemplo, el elemento BODY es un nodo hijo de HTML (nodo padre); el elemento párrafo, representado por la etiqueta <P>, por lo general es hijo de BODY (nodo padre), etc. Cuando un elemento no contiene otros elementos, se dice que es un nodo final (las hojas de un árbol) y su contenido suele ser generalmente texto como en el caso de los títulos<H1>,<H2>... o un elemento vacío <HR>,
.

DOM se utiliza, por lo tanto, para definir la estructura de un documento HTML. Una vez que obtenemos la estructura con todos los elementos existentes, es posible acceder a ellos individualmente, a sus métodos y propiedades y, por lo tanto, programarlos utilizando un lenguaje de programación de *script*. A continuación veremos cómo hacerlo utilizando JavaScript.

Hemos de tener en cuenta que, para poder acceder más cómodamente al elemento (objeto) que deseamos programar, hemos de asignarle un nombre o identificador mediante uno de los atributos comunes NAME o ID, respectivamente.

A continuación veremos un ejemplo de cómo acceder a un formulario sencillo y obtener el valor de una entrada de texto. Al formulario le pondremos por nombre (atributo NAME) "form1" y a la entrada de texto, "entrada", es decir:

```
<FORM NAME="form1">

<INPUT TYPE="TEXT" NAME="entrada" VALUE="">

</FORM>
```

Para acceder al valor de la entrada de texto utilizando el modelo DOM, tendríamos que utilizar la siguiente estructura de nodos:

document.form1.entrada.value

Es decir, partimos del nodo raíz (objeto *document*), accedemos al nodo hijo formulario (objeto *form1*) y desde éste a su nodo hijo entrada de texto (objeto entrada). Una vez que hemos accedido al objeto entrada, obtenemos su valor (propiedad *value*), es decir, el texto existente en la entrada del texto del formulario.

No es necesario que los objetos de DOM que son colecciones (listas de un determinado objeto) tengan un nombre o identificador y se puede acceder a ellos por su índice. Éste es el caso de la colección formularios (*forms*). Debemos tener en cuenta que el índice de las colecciones comienza en cero [0] y va entre corchetes. Por ello también podríamos haber accedido al valor de la entrada de texto con la siguiente ruta de nodos:

document.forms[0].entrada.value

De hecho, se puede acceder a cualquier elemento del documento por su índice utilizando la colección all. Como comprenderá, utilizar este método es más complicado que el asignar un determinado nombre o identificador al elemento que queremos acceder para obtener su valor o programar.

Continuando con lo expuesto anteriormente sobre cómo acceder al valor de la entrada de texto, podríamos escribir una función en JavaScript que nos muestre el dato existente en un cuadro de diálogo, por ejemplo, la siguiente:

```
function diNombre(){
    var valor = "Nombre introducido: ";
    //valor += document.forms[0].entrada.value;
    valor += document.form1.entrada.value;
    alert( valor );
}
```

También añadiremos al formulario un par de elementos más, una etiqueta explicativa y un botón que, al ser pulsado, ejecute la función diNombre():

```
<FORM NAME="form1">
<LABEL FOR="entrada">Nombre:</LABEL>
<INPUT TYPE="TEXT" NAME="entrada" VALUE="">
<BUTTON onClick="diNombre();">Obtener nombre</BUTTON>
</FORM>
```

Listado completo (***CD-ROM*** *directorio* ***Ejemplos\Cap03\domhtml.htm***)

domhtml.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
```

```
<HTML>
<HEAD>
<TITLE> DOM HTML y JavaScript </TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function diNombre(){
    var valor = "Nombre introducido: ";
    //valor += document.forms[0].entrada.value;
    valor += document.form1.entrada.value;
    alert( valor );
}
-->
</SCRIPT>
</HEAD>
<BODY>
<DIV STYLE="text-align:center">
<H3> Programación DOM-HTML con JavaScript </H3>
<FORM NAME="form1">
 <LABEL FOR="entrada">Nombre:</LABEL>
 <INPUT TYPE="TEXT" NAME="entrada" VALUE="">
 <BUTTON onClick="diNombre();">Obtener nombre</BUTTON>
</FORM>
</DIV>
</BODY>
</HTML>
```

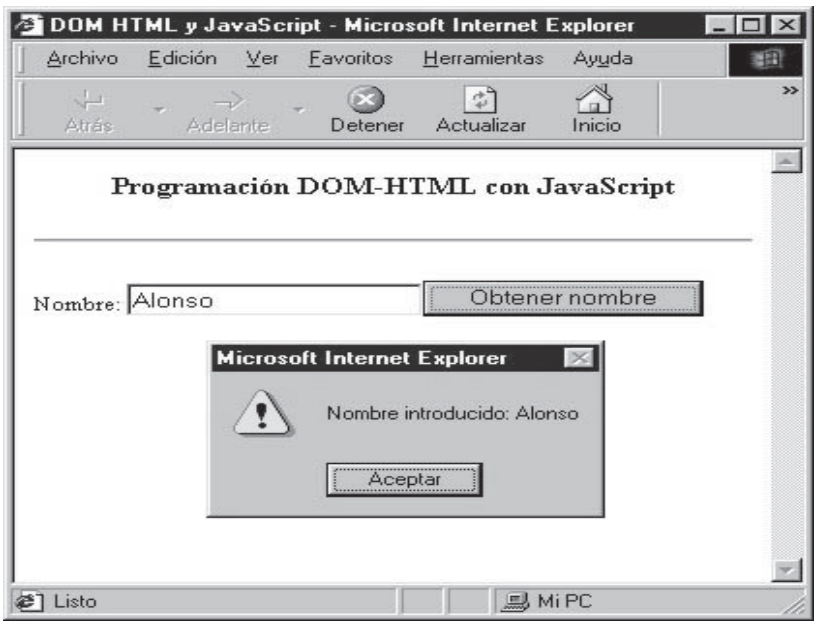

Figura 3 - 6. Ejemplo de programación con DOM

LOS PRINCIPALES OBJETOS DE DOM

• **window**: por medio de sus propiedades y métodos se puede acceder a las características del navegador y los documentos abiertos en él.

• **document**: sus propiedades y métodos nos permiten acceder a todos los elementos utilizados en el documento abierto.

EL OBJETO *WINDOW*

Como hemos visto en la representación gráfica de DOM de HTML, es el objeto raíz del cual penden todos los demás objetos DOM. Concretamente, son siete objetos:

• **document** el objeto documento, ya ampliamente explicado.

• **frames** documentos divididos en varias ventanas y con varios documentos

• **history** posibilita la navegación por las páginas visitadas, avanzando o retorciendo por ellas.

• **location** objeto con propiedades y métodos diseñados para la manipulación de direcciones URL.

• **event** objeto basado en el modelo de eventos de DHTML por medio del cual podemos acceder a los sucesos ocurridos en un documento o en sus elementos individuales. Los eventos pueden ser producidos por el sistema (carga de una página...), el usuario (pulsación del botón del ratón...) o cualquier otro mecanismo (un temporizador...). La respuesta a determinados eventos y para algunos elementos en concreto pueden estar ya programados con acciones predeterminadas, como, por ejemplo, el abrir una página cuando se pulsa sobre un hiperenlace, elemento ancla representado por la etiqueta <A>. No obstante, mediante programación podemos modificar esa respuesta o, incluso, anular todas ellas con una instrucción similar a <BODY onClick="return false">.

• **screen** objeto con información sobre la pantalla del ordenador cliente.

De todos ellos, los utilizados para la programación DOM de los documentos HTML y sus contenidos son **event** y **document**.

PROGRAMACIÓN POR MEDIO DEL OBJETO *EVENT*

El principal objeto relacionado con la programación de los elementos existentes en un documento es **event**, ya que proporciona una serie de propiedades y métodos por

medio de los cuales programar el comportamiento del objeto que produce el suceso. Entre estas propiedades tenemos ***srcElement*** por medio de la cual podemos acceder al elemento fuente que ha producido el evento y a partir de él, por lo tanto, a las propiedades y métodos de los elementos (objetos de tipo *element*) existentes en el documento. Según esto sería posible cambiar de forma dinámica el color de un determinado párrafo al situar el cursor sobre él utilizando la propiedad *style* con una instrucción similar a:

```
<P STYLE="font-family: Arial; color: red"
onMouseOver="window.event.srcElement.style.color='blue"
onMouseOut="window.event.srcElement.style.color='red'"> Cambiar de color
</P>
```

Como podemos observar, ésta es otra forma de programación sin utilización de un determinado tipo de *scripts*, sino accediendo directamente a las propiedades de los objetos generados por DOM.

De forma similar y utilizando la propiedad className, correspondiente al atributo CLASS, podríamos utilizar o cambiar una determinada clase de estilo definida en un bloque <STYLE>. Así, por ejemplo, si tuviésemos dos clases:

```
<STYLE>
.textoAzul { color: blue }
.textoRojo { color: red }
</STYLE>
```

Podríamos efectuar el cambio con la instrucción:

```
<P STYLE="font-family: Arial; color: red"
onMouseOver="window.event.srcElement.className= 'textoAzul' "
onMouseOut="window.event.srcElement.className='texto Rojo'"> Cambiar de color
</P>
```

PROGRAMACIÓN POR MEDIO DEL OBJETO *DOCUMENT*

También es posible acceder a elementos en concreto existentes en un documento utilizando algunas de las siguientes propiedades o métodos.

Propiedades

- **document.activeElement.innerHTML**: muestra el contenido de cualquier elemento.

- **document.activeElement.outerHTML**: muestra el contenido de cualquier elemento *tag* incluido.

• **document.activeElement.innerTEXT**: muestra el texto de cualquier elemento.

• **document.documentElement.innerHTML**: muestra el texto de cualquier elemento.

• **document.ocumentElement.outerHTML**: muestra todo el texto del documento.

• **document.ocumentElement.innerTEXT**: muestra el texto de cualquier elemento.

Métodos

- **Para comprobar el número de elementos:**

  document.GetElementsByID("id").length

  document.GetElementsByName("name").length

  document.GetElementsByTagName("tag").length

- **Para mostrar datos de un elemento:**

  document.GetElementsByName("name").innerHTML

Los métodos y propiedades vistos anteriormente corresponden al DOM HTML, los propiamente DOM están definidos en la denominada interfaz DOM Core.

EL INTERFAZ DOM CORE

Establece las propiedades y métodos específicos de DOM (DOM Core) y es el utilizado para la programación de documentos basados en XML. Los contenidos de una página HTML también pueden ser programados utilizando este interfaz. A modo de ejemplo, a continuación se muestra cómo crear una lista de forma dinámica utilizando los métodos propios de este interfaz:

domhtmlJavaScript.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD>
<TITLE>DOM HTML JavaScript</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!-- Comienzo de ocultación a antiguos browsers
```

```
function creaLista(){
    oLista = document.createElement("OL");
    oItem = document.createElement("LI");
    oItem2 = oItem.cloneNode();
    oItem3 = oItem.cloneNode();
    oItem4 = oItem.cloneNode();
    texto1 = document.createTextNode("Item 1");
    texto2 = document.createTextNode("Item 2");
    texto3 = document.createTextNode("Item 3");
    texto4 = document.createTextNode("Item 4");
    oItem.insertBefore( texto1 );
    oItem2.insertBefore( texto2 );
    oItem3.insertBefore( texto3 );
    oItem4.insertBefore( texto4 );
    oLista.appendChild( oItem );
    oLista.appendChild( oItem2 );
    oLista.appendChild( oItem3 );
    oLista.appendChild( oItem4 );
    nodoBody.appendChild( oLista );
}
// Final de la ocultación -->
</SCRIPT>
</HEAD>
<BODY ID="nodoBody">
<BUTTON onClick="creaLista()">Crear lista dom</BUTTON>
</BODY>
</HTML>
```

Utilizando VBScript, métodos de DOM de HTML y métodos de DOM Core, podríamos efectuar la siguiente implementación para la subrutina encargada de crear la lista dinámicamente:

```
<SCRIPT LANGUAGE="VBScript">
Sub creaLista()
    Dim oLista, oItem, oItem2, oItem3, oItem4
    Set oLista = document.CreateElement("OL")
    Set oItem = document.CreateElement("LI")
    Set oItem2 = oitem.cloneNode()
    Set oItem3 = oitem.cloneNode()
    Set oItem4 = oitem.cloneNode()
    oLista.appendChild( oItem )
    oLista.appendChild( oItem2 )
    oLista.appendChild( oItem3 )
    oLista.appendChild( oItem4 )
    oItem.innerHTML = "item 1"
    oItem2.innerHTML = "item 2"
    oItem3.innerHTML = "item 3"
    oItem4.innerHTML = "item 4"
    document.body.appendChild( oLista )
```

```
End Sub
// Final de la ocultación -->
</SCRIPT>
```

CONCLUSIÓN

El HTML Dinámico es muy complejo y puede utilizar diversas tecnologías. Este capítulo nos ha servido para tomar contacto con algunas de ellas que serán utilizadas en capítulos posteriores, como son las hojas de estilo.

Como se comprenderá, el profundizar en todo lo tratado en este capítulo se hace imposible y se muestran los aspectos más relevantes o las nociones básicas que nos permitan comprender el funcionamiento de esas tecnologías. En el próximo capítulo veremos más a fondo las hojas de estilo CSS, tecnología base para la creación de estilos personalizados utilizados tanto por HTML como por XML.

COMPLEMENTOS

En el CD-ROM

dom\DOM.zip Archivo comprimido en formato zip con las especificaciones de DOM proporcionado por el W3C.

dom\dom1-es.zip Archivo comprimido en formato zip con las especificaciones de DOM traducidas al español por Juan R. Pozo.

dom\DOMLevel-1\cover.html Recomendaciones para DOM Level-1 del W3C en formato HTML.

dom\dom1-es\cover.html Traducción al español de las recomendaciones DOM en formato HTML.

html \html40\html40.pdf Archivo PDF con las especificaciones de HTML 4.0 del W3C.

html \html40\html401-es.pdf Archivo PDF con las especificaciones de HTML 4.01 traducidas al español por Juan R. Pozo.

html\html40\html401.zip Archivo comprimido con las especificaciones de HTML 4.01, el documento original del W3C en formato HTML.

html\html40\html401 Directorio con los archivos descomprimidos en formato HTML del anterior. Se accede al índice abriendo el archivo "**cover.html**".

html\html40\html401-es.zip Archivo comprimido con las especificaciones de HTML 4.01; traducción al español de Juan R. Pozo.

html\Html40\Html401-es Directorio con los archivos descomprimidos en formato HTML del anterior. Se accede al índice abriendo el archivo "**cover.html**".

html\html40\html40hlp.zip Archivo comprimido con las especificaciones de HTML 4.0 en formato *help* de Windows de Web Design Group (WDG).

html\html40\html40hlp Directorio con los archivos descomprimidos en formato HELP de Windows del anterior. El documento de ayuda WDG HTML 4.0 Reference se abre desde el archivo "**index.hlp**".

html\validador\cselite Utilidad de validación HTML "CSE HTML Validator Lite".

En Internet

http://www.w3.org/TR/REC-html40 Recomendaciones de HTML 4.0 en W3C.

http://www.w3.org/TR/html401 Especificaciones de HTML 4.01 en W3C.

http://www.htmlhelp.com/ Web Design Group (WDG), Documentación HTML 4.0.

http://www.htmlhelp.com/tools/validator/ WDG HTML Validator.

http://validator.w3.org/ Servicio de validación de HTML y XHTML de W3C.

http://www.w3.org/TR/REC-DOM-Level-1 Recomendaciones DOM de W3C.

http://www.htmlvalidator.com Herramientas de validación HTML.

http://www.zvon.org/ Referencias con ejemplos y tutoriales.

http://www.zvon.org/xxl/JSDOMFactory/index.html JavaScript&DOM_Factory.

http://dns.uncor.edu:80/info/tutor_sp/tutorial.htm Tutorial de HTML en español.

http://www.sidar.org/traduc/traduc.htm SID@R Traducciones, documentación y referencias traducidas al español.

http://html.conclase.net/ HTML con Clase, lugar mantenido por Juan R. Pozo. Tutoriales, documentación y traducciones de referencias y especificaciones.

CAPÍTULO 4

HOJAS DE ESTILO EN CASCADA

INTRODUCCIÓN

En este capítulo estudiaremos una de las tecnologías más interesantes relacionadas con la publicación de documentos en Internet: la aplicación de estilos personalizados a nuestros documentos por medio del uso de las denominadas hojas de estilo en cascada.

En el capítulo anterior habíamos utilizado esta técnica de su forma más simple por medio del atributo común STYLE aunque no nos habíamos detenido a explicar en profundidad las propiedades utilizadas ni cómo se definían y usaban. El objetivo del presente capítulo es enseñarle todas las propiedades existentes en las hojas de estilo para que al final de él sepa definir sus propios estilos y utilizarlos en los documentos HTML; en un próximo capítulo aprenderá a cómo utilizarlos para publicar documentos basados en XML.

CSS

Cascading Style Sheets, ***Hojas de Estilo en Cascada*** es un estándar nacido en el año 1996 y utilizado para la aplicación de estilos a los documentos publicados en Internet. Esta tecnología se suele utilizar combinada con los lenguajes de marcas HTML, XHTML, XML, etc., para superar sus limitaciones en la definición de estilos.

Existen dos niveles o recomendaciones: CSS1 (1996) y CSS2 (1998); los navegadores actuales solamente dan soporte a CSS1. Las hojas de estilo son soportadas por los navegadores Internet Explorer y Netscape Navigator desde su versión 3. No obstante, las implementaciones y, por lo tanto, la visualización de los documentos que utilizan hojas de estilos puede variar de uno a otro.

Las Recomendaciones para CSS del W3C se encuentran en los siguientes documentos:

• **Recomendaciones para las hojas de estilo en cascada nivel 1** del 17 de diciembre de 1996.

REC-CSS1-19990111
Cascading Style Sheets, level 1
W3C Recommendation 17 Dec 1996
http://www.w3.org/TR/1999/REC-CSS1

• **Especificaciones para las hojas de estilo en cascada nivel 2** del 12 de mayo de 1998.

REC-CSS2-19980512
Cascading Style Sheets, level 2 CSS2 Specification
W3C Recommendation 12-May-1998
http://www.w3.org/TR/REC-CSS2

Quienes no tengan posibilidad de acceso a Internet pueden encontrar esos documentos en el ***CD-ROM directorios css\Css1*** *y* ***css\Css2.*** Las traducciones al español se encuentran en los ***directorios css\Css1(sp)*** *y* ***css\Css2(sp).*** También se incluye "WDG's CSS Reference", un archivo en formato hlp (ayuda de Windows) de Web Design Group, de libre distribución, en el directorio ***css\CSShelp\css.hlp.***

Actualmente existe un borrador de trabajo para las hojas de estilo nivel 3 o CSS3; si lo desea, puede consultar ese documento con título W3C Working Draft, 23 May 2001 en la dirección http://www.w3.org/TR/css3-roadmap.

HTML Y CSS

Aunque HTML es un lenguaje claramente enfocado a la composición o maquetación y formato de documentos para la Web, carece de elementos propios para la definición de estilos. Como ya vimos en los anteriores capítulos, la única etiqueta que nos permite actuar sobre los estilos predeterminados es <FONT> y, según las recomendaciones, solamente es posible el cambio del color y tamaño por medio de sus atributos COLOR y SIZE. Por otra parte, el tamaño solamente puede ser modificado en un rango cuyos valores van de -7 a +7, siendo el valor preestablecido 5. Para superar estas limitaciones, algunos navegadores implementaron soporte para el atributo FACE y por medio de él se puede seleccionar también el tipo de fuente de letra que deseamos utilizar: Arial, Curier, etc., que ya habíamos utilizado para dar formato a nuestro documento en el capítulo 2.