Edición electrónica

Ricardo Albiñana Bertomeu
Cristina Fuertes Royo
Xavier Sanchez Porras

PID_00201874

UOC
Universitat Oberta de Catalunya

Índice

Introducción

El primer apartado de este módulo se centra en los lenguajes de marcado como nuevo concepto en la edición de textos electrónicos y se ofrece un amplio abanico de este tipo de lenguaje. De manera general, se tratan, tanto jerárquica como extensivamente, aquellos lenguajes de marcado que intervienen o podrían intervenir de una u otra manera en el diseño y la implementación de páginas web.

Se presenta SGML como un metalenguaje que permite la creación de distintas organizaciones de marcado, HTML como lenguaje base de la WWW y vehículo de la creación de páginas web y de la parte "visible" de Internet, y XML como lenguaje extendido y capaz de otorgar una semántica a las estructuras de datos con la que el software los podrá gestionar con cierta inteligencia, así como la unión de los dos últimos (XML y HTML) en XHTML.

Asimismo, se lleva a cabo una introducción a las DTD, aspecto complejo del que tan sólo se plantearán las nociones preliminares. También se presenta el concepto DHTML, sobre todo para distinguirlo de los anteriores, pues no se trata de un lenguaje específico, sino de una combinación de tecnologías muy particular, que si bien tienen que ver con la creación de webs, no suponen un lenguaje de marcado distinto en sí mismo.

Se plantea el desarrollo de XSL y XSLT, aunque se han incluido de manera somera, porque a poco que el estudiante profundice en el desarrollo interno de la WWW a partir de lenguajes de marcado, podrá adquirir conocimientos avanzados de este punto.

En el segundo apartado, dedicado a HTML y XHTML, se realiza una introducción al lenguaje de marcas HTML diseñado para estructurar los documentos de texto y relacionarlos a modo de hipertexto en páginas web. Mediante un breve recorrido histórico por las diferentes versiones del lenguaje se verá la evolución y los cambios más relevantes. Se tratará con cierta exhaustividad el lenguaje HTML 4.01 por ser el estándar más aceptado en el momento de redactar este material.

Respecto a la creación de documentos, se analizarán los principios generales que recomienda el W3C.

Una parte importante de este apartado se centra en conocer la estructura básica de un documento web y las etiquetas principales: el concepto de etiqueta, el encabezado y el cuerpo. Se analizarán sus elementos y atributos principales.

Dentro de este mismo apartado se incluye también el estándar de escritura XHTML 1.0 y sus requisitos básicos con relación a la separación del contenido y la apariencia de los documentos.

Otro de los aspectos que se aborda en este apartado es la validación del HTML 4.01 y el XHTML 1.0, comentando las herramientas básicas de validación que se pueden encontrar en la Web. Asimismo, se realiza una pequeña introducción al lenguaje HTML5 y XHTML5 con la incorporación de sus nuevos elementos y mejoras.

En el tercer apartado se describen las posibilidades, las fuentes y el uso de las hojas de estilo en cascada (HEC) o *cascade style sheets* (CSS), muy importantes a la hora de otorgar el aspecto visual, e incluso en ocasiones interactivo, de una página web. Conoceremos los elementos mínimos para poder programar con CSS y haremos un repaso a las etiquetas básicas de este lenguaje orientado el estilo y a la visualización de los elementos de una página HTML.

Veremos los distintos modos de incardinar los datos de las hojas CSS, bien en el mismo archivo HTML, bien en hojas separadas, de manera que este estilo pueda ser reutilizado con el mínimo esfuerzo.

Una vez puestos al día con los elementos básicos, pasaremos a realizar unas pequeñas prácticas que nos familiarizarán con los trucos más básicos que deberemos conocer a la hora de aplicar CSS en nuestros documentos HTML. Finalmente elaboraremos un breve documento con CSS.

En el cuarto apartado se analizan las herramientas de diseño y navegación. Respecto a los editores web, se verán las características básicas de los editores WYSIWYG, el control del código del diseño y las diferencias entre la vista de diseño y los diferentes navegadores web.

Asimismo, se introducirán las principales características y opciones de distintos editores web: herramientas básicas como Notepad; herramientas de código libre como NVU, Komposer y Amaya, y herramientas de autor como Dreamweaver.

El apartado continuará con la presentación de los principales navegadores web que un usuario puede utilizar: Internet Explorer en sus diferentes versiones, Firefox de Mozilla y sus complementos, Operam Safari y Goggle Crome.

Por último, se introducirán nociones sobre la arquitectura cliente-servidor y la vida de una página web.

En el quinto apartado se trabajarán las técnicas de maquetación de un documento, se presentarán la estructura y el aspecto del conjunto de la página web y se hará un poco de historia acerca de lo que ha sido la maquetación a lo largo de los últimos tiempos de la WWW.

Asimismo, se comentarán los distintos modos de estructurar y organizar el aspecto general de un documento web desde la perspectiva de las tablas, los marcos y la etiqueta DIV manipulada desde o en conjunto con las CSS.

El sexto apartado se centrará en los elementos avanzados: un conjunto de tecnologías que suelen trabajar en coordinación o en el interior del HTML. Se introducirán nociones muy básicas del funcionamiento de JavaScript con el objetivo de que el estudiante sea capaz de comprender sus mecanismos. Mediante enlaces e indicaciones, el programador novel será capaz de dominar el uso de JavaScript por sus propias prácticas y lecturas.

Además, se presentarán las tecnologías subyacentes en DHTML y se llevará a cabo una aproximación al lenguaje de programación JAVA, con los enlaces e indicaciones necesarias para poder llegar a aprender y practicar este lenguaje. Sobre Flash se realizará una breve introducción y se darán indicaciones de cómo obtener el software para el desarrollo de aplicaciones con esta herramienta. Y, por último, se presentará PHP y se aportarán algunos pequeños ejemplos de su funcionamiento para que el estudiante tenga los instrumentos que le permitan iniciarse en este tipo de programación.

En el séptimo apartado se proporcionan los protocolos de actuación y métodos necesarios para la evaluación de recursos digitales y fuentes de información de calidad publicadas en Internet. El objetivo principal de este apartado se centra en el análisis de los principales criterios que deben tenerse en cuenta para evaluar una página web. Se proporcionarán las herramientas para su análisis y validación, las plantillas y los elementos imprescindibles para su catalogación. Y también se definen las finalidades y los objetivos de las agencias de evaluación, los organismos encargados de la búsqueda, selección y descripción de recursos digitales en la Web.

En el octavo apartado se analizarán los principios de diseño para una arquitectura de la información eficiente en sitios web. El objetivo principal es el examen de metodologías para un diseño estructural eficaz de sistemas de organización, recuperación y estructuración de contenidos, especialmente los que se implementan con tecnología web.

El módulo finalizará con dos estudios, uno sobre usabilidad, disciplina que estudia el modo de diseñar sitios web para que los usuarios puedan interactuar entre sí de la manera más fácil, cómoda e intuitiva posible, y otro sobre accesi-

bilidad, un procedimiento que permite que un producto o servicio web pueda ser utilizado por el mayor número de personas posible, sin tener en cuenta las limitaciones propias del individuo o de las derivadas del contexto de uso.

Objetivos

Los objetivos que el estudiante tiene que alcanzar una vez trabajados los contenidos de este módulo son los siguientes:

1. Adquirir los conocimientos necesarios sobre la edición electrónica en web y su implementación, mediante la comprensión de conceptos básicos sobre el entorno multimedia, el hipertexto y la hipermedia.

2. Adquirir conocimientos mínimos de HTML para profundizar en aspectos fundamentales y concretos que hacen referencia a los diferentes niveles de estructuración de la información entorno a los hipertextos.

3. Conocer los diferentes lenguajes de marcaje con el objetivo de entender las posibilidades creativas que nos ofrecen si queremos publicar documentos digitales en Internet.

4. Iniciarse en la utilización de elementos HTML avanzados desarrollados en los materiales de la asignatura, como las hojas de estilo en cascada.

5. Introducir la evaluación de publicaciones digitales.

6. Asimilar los conceptos de usabilidad y accesibilidad de un software.

1. SGML y lenguajes ML. Lenguaje de etiquetado de documentos digitales

1.1. Objetivos

Este apartado tiene como objetivo conocer los distintos tipos de lenguajes de etiquetado de documentos derivados del SGML, así como analizar sus características y prestaciones.

1.2. Introducción

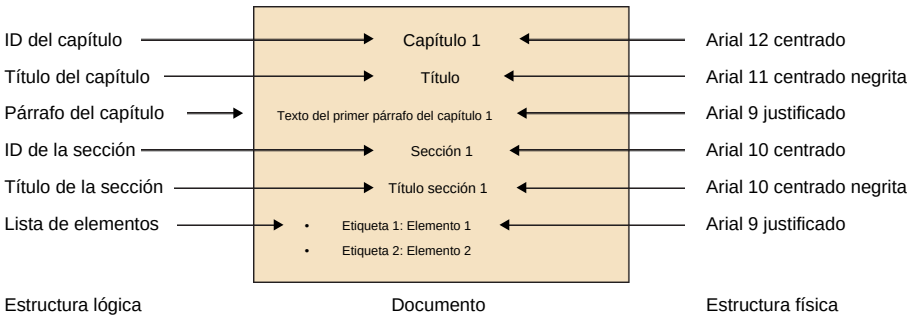
Tres de los componentes esenciales de la World Wide Web son:

- El lenguaje de descripción de páginas web **HTML** (*hypertext mark-up language*).
- El **HTTP** (*hypertext transfer protocol*), o protocolo de transferencia de un hipertexto que mantiene el esquema de petición-respuesta entre un cliente y un servidor, permitiendo, por ejemplo, la visualización de páginas web siguiendo los enlaces incorporados en éstas.
- El **navegador** es el intérprete de la presentación (aspecto) de los distintos elementos marcados que componen un documento.

La mayoría de los documentos electrónicos que utilizamos normalmente contienen dos tipos de información:

- El contenido del texto del propio documento.
- Otra información complementaria (metainformación), un juego de códigos (también denominados etiquetas) que proporciona la información sobre cómo mostrar o interpretar el texto. Estos códigos son el **marcado**.

Estructura física y lógica de un documento



Con el desarrollo de Internet han aparecido una serie de lenguajes para la creación de contenidos que facilitan el acceso a la información con independencia de la plataforma o el sistema informático utilizado.

Por **lenguaje de marcas** nos referimos a un conjunto de convenciones de marcas utilizado para la codificación e interpretación de un texto que desarrolla la idea de separar presentación y estructura en un documento electrónico.

Origen de los lenguajes de marcado

El origen de los actuales lenguajes de marcado fue el *generalized markup language* o GML, un proyecto de la empresa IBM para mantener grandes cantidades de documentos.

C. F. Goldfarb (1990). *The SGML Handbook*. Oxford University Press.

El World Wide Web Consortium (W3C) es un consorcio internacional que coordina y desarrolla las recomendaciones para la World Wide Web (http://www.w3.org).

1.3. SGML

El lenguaje de marcado generalizado estándar **SGML** (*standardized generalized markup language*) es un metalenguaje, es decir, un medio de describir formalmente lenguajes de marcado.

Es un estándar internacional publicado en el año 1986 (ISO 8879). Esta norma recoge las bases teóricas de una serie de aplicaciones de las que la más extendida es el **HTML**.

Lectura recomendada

International Organization for Standardization (1986). *ISO 8879: Information processing - Text and office systems - Standard Generalized Markup Language (SGML)*. Ginebra: ISO.

El SGML proporciona esquemas de marcado con una sintaxis para definir los elementos de datos que forman un documento electrónico, cuál es su estructura y cómo deben marcarse, para que puedan ser tratados de manera automática.

Especifica la sintaxis para la inserción de marcas en los textos, así como la sintaxis del documento que detalla qué etiquetas están admitidas y dónde. La definición de la estructura y el contenido de un documento se realizan por medio de su **DTD** (*document type definition*).

Ved también

Podéis ver el apartado 1.7, "DTD".

SGML no proporciona un conjunto fijo de marcas o de tipos de documentos, sino un método formal y general para diseñar clases de documentos adecuadas a cualquier situación o necesidad específica.

Ventajas e inconvenientes de SGML

- Flexibilidad, integridad, portabilidad, mayor control y reutilización de la información.
- La potencia de SGML implica gran complejidad para su aprendizaje y programación.

Webs recomendadas

W3C Overview of SGML Resources. http://www.w3.org/MarkUp/SGML/

Desde este enlace podéis consultar las aplicaciones y principales iniciativas en el mundo académico del SGML: http://xml.coverpages.org/acadapps.html

1.4. HTML

Creado por Tim Berners-Lee a principios de los años noventa, su origen es el Laboratorio Europeo de Física de Partículas (CERN), cuyo objetivo principal fue disponer de un sistema de distribución y transmisión de documentos para el mundo científico.

HTML (*hypertext markup language* o lenguaje de marcas de hipertexto) ha desempeñado un papel fundamental en el crecimiento de Internet, ya que es lenguaje de marcado más usado para la realización de páginas web.

En HTML los elementos como imágenes, símbolos, sonido, vídeo, etc. no están almacenados en la página, sino que se insertan desde otro fichero.

Es una aplicación de la norma ISO 8879 (SGML) con una definición del tipo de documento (DTD) preestablecido. En el DTD figuran las declaraciones de los elementos indicando el nombre oficial de cada etiqueta y lo que cada una de ellas puede contener.

Por lo tanto, los documentos HTML son documentos de texto en código ASCII (texto plano), con una serie de etiquetas que definen una serie de características que el navegador interpreta y representa en la pantalla del ordenador.

Ventajas e inconvenientes de HTML

- Presenta limitaciones relacionadas con el tratamiento de información dinámica, ya que dispone de un número fijo de etiquetas.
- Es muy simple y sencillo de aprender y usar.
- No requiere herramientas especiales y está muy difundido.
- Está orientado fundamentalmente a la representación de los datos y no a su estructura. No es extensible y no permite la reutilización de la información.

Web recomendada

HTML. http://www.w3.org/html/

Ved también

Para ver con más detalle este tema, podéis ver el apartado 2, "HTML y XHTML".

1.5. XML

XML (*extensible markup language*) surge en 1996 en el World Wide Web Consortium como respuesta a la falta de organización que supuso el rápido crecimiento del HTML. Es un lenguaje estructurado, con un vocabulario extensible y que se puede validar, lo que permite la transmisión de información estructurada.

Es una manera restringida de SGML, optimizada para su utilización en Internet, tomando de éste sus ventajas y restándole complejidad, ya que es simple de usar y se basa en etiquetas de texto.

Webs recomendadas

XML Technology (W3C). http://www.w3.org/standards/xml/

Extensible markup language (XML) 1.0 - El lenguaje extensible de marcas (XML). http://www.sidar.org/recur/desdi/traduc/es/xml/xml1/index.html

XML permite el acceso a documentos desde cualquier dispositivo. Se orienta a los datos, a su semántica y no a la representación, por ello se está convirtiendo en el lenguaje utilizado por las **bases de datos** en la Web, además de porque permite un fácil intercambio de información entre aplicaciones.

Diferencia entre XML (estructura) y HTML (presentación):

Ejemplo: XML

```
<pelicula>
<titulo>Casablanca</titulo>
<actor>Humphrey Bogart</actor>
<actriz>Ingrid Bergman</actriz>
<director>Michael Curtiz</director>
<compañía>Warner Bros. Pictures</compañía>
<duracion>102</duracion>
<año>1946</año>
</pelicula>
```

Ejemplo: HTML

```
<html>
<body>
<h3>Casablanca</h3>
<p>Actores: Humphrey Bogart e Ingrid Bergman</p>
<ul>
<li><p>director: Michael Curtiz</p></li>
<li><p>compañía: Warner Bros. Pictures</p></li>
<li><p>duración: 102</p></li>
<li><p>año: 1946</p></li>
</ul>
</body>
</html>
```

1.6. XHTML

Es un **lenguaje de descripción** para diseñar páginas web. No es un lenguaje de programación, sino de marcado.

El objetivo del W3C, su creador, fue lograr una web semántica en la que la estructura y los datos estén separados de una manera clara, donde la apariencia visual y el aspecto final del documento se representan mediante hojas de estilo (CSS), y no por etiquetas como en el HTML.

El lenguaje **XHTML** se basa en el código ASCII o ANSI (texto plano), no contempla una interfaz determinada para su tratamiento, lo que significa que cualquier procesador de texto puede utilizarse para la realización de páginas web, como el simple bloc de notas de Windows.

Ved también

Para ver con más detalle este tema, podéis ver el apartado 2, "HTML y XHTML".

Web recomendada

XHTML Working Group Home Page (W3C). http://www.w3.org/MarkUp/

1.7. DTD

Una **definición de tipo de documento** o **DTD**, siglas en inglés de *document type definition*, es una descripción de estructura y sintaxis de un documento XML o SGML. Su función básica es la descripción del formato de datos.

"En el centro de una aplicación SGML hay un fichero llamado DTD o Definición de Tipo de Documento. La DTD describe la estructura de un documento como un esquema de bases de datos describe los tipos de información que trata y las relaciones de campos. Así pues, una DTD proporciona una estructura para los elementos que constituyen un documento (por ejemplo, capítulos, encabezamientos, secciones, materias...); y también especifica las reglas para las relaciones entre elementos. Estas reglas ayudan a asegurar que los documentos tengan una estructura coherente y lógica".

Lectura recomendada

A. Delgado Gómez (2003). *Normalización de la descripción archivística: Introducción a Encoded Archival Description* (EAD).

El **DTD** define lo que significa exactamente cada una de las marcas contenidas en un documento para identificar sus datos, ya que cuando, por ejemplo, hace referencia a un documento XML, éste debe comportarse correctamente en función de las normas del DTD incluidas.

DTD

El navegador web identifica el DTD (*document type definition*) de un documento XHTML y verifica que sea sintácticamente correcto.

Volviendo al ejemplo anterior del documento XML (películas.xml):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<PELICULAS>
<pelicula>
<titulo>Casablanca</titulo>
<actor>Humphrey Bogart</actor>
<actriz>Ingrid Bergman</actriz>
<director>Michael Curtiz</director>
<compañia>Warner Bros. Pictures</compañia>
<duracion>102</duracion>
<año>1946</año>
</pelicula>
<PELICULAS>
```

Su **DTD** sería (peliculas.dtd):

```
<!ELEMENT peliculas (pelicula)+>
<!ELEMENT pelicula (título,actor,actriz,director,compañía,duración,año)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT actor (#PCDATA)>
<!ELEMENT actriz (#PCDATA)>
<!ELEMENT director (#PCDATA)>
<!ELEMENT compañia (#PCDATA)>
<!ELEMENT duracion (#PCDATA)>
<!ELEMENT año (#PCDATA)>
```

Un documento se relaciona con su DTD en la declaración de tipo de documento (DOCTYPE), es decir, una línea en el fichero XML que haga referencia al fichero, en este caso, peliculas.dtd.

Web recomendada

W3C XML Specification DTD. http://www.w3.org/XML/1998/06/xmls-pec-report-19980910.htm

1.8. DHTML

DHTML (*dynamic HTML* o HTML dinámico) posibilita la creación de páginas web interactivas sin las limitaciones del HTML.

DHTML no es en sí mismo un lenguaje de programación, ni una aplicación, ni un estándar, sino que engloba muchas técnicas y programas distintos, se trata de una metodología de trabajo, una combinación de:

a) HTML;
b) hojas de estilo en cascada (CSS);
c) un lenguaje de *script* (JS [JavaScript], lenguaje de programación de *scripts*);
d) capas (*layers*), y
e) DOM (*document object model*)

DOM

Modelo en objetos para la representación de documentos es una plataforma y lenguaje que permite a los programas y *scripts* acceder y actualizar dinámicamente el contenido, la estructura y el estilo de los documentos (http://www.w3.org/DOM/).

Criterio común

Aunque hay que destacar que en la actualidad no existe un criterio común respecto a este lenguaje entre los principales navegadores del mercado.

Ejemplo

Efectos distintos en las páginas, ayudas interactivas, sonidos, movimiento de objetos, vídeos, menús interactivos, control y respuesta a las acciones de un usuario en la página, formularios y cuestionarios dinámicos, control sobre los formularios, etc.

Web recomendada

Web Design and Applications (W3C). http://www.w3.org/standards/webdesign/

Ved también

Para ver con más detalle este tema, podéis ver el apartado 6, "Elementos avanzados".

1.9. CSS

Las **hojas de estilo en cascada** (en inglés *cascading style sheets*, **CSS**) es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los navegadores.

Ved también

Para ver con más detalle este tema, podéis ver el apartado 3, "Hojas de estilo en cascada (CSS)".

Webs recomendadas

Cascading Style Sheets (W3C). http://www.w3.org/Style/CSS/

W3C. http://www.w3.org/Style/. Traducción al español: http://www.spanish-translator-services.com/espanol/t/Style/.

Lectura recomendada

E. Castro (2003). *HTML con XHTML y CSS*. Madrid: Anaya Multimedia. ISBN: 84-415-1533-6.

Cuando trabajamos con gran cantidad de documentos en un sitio web, nos interesa que todos tengan los mismos estilos y propiedades. Un procedimiento sería repetir el fragmento de código en todos ellos, aunque en el momento en el que decidamos cambiar un estilo, estaremos obligados a modificar todos los documentos. Otro procedimiento es utilizar una hoja de estilo, es decir, un archivo independiente referenciado mediante un enlace en cada documento que contiene todas las especificaciones, con lo que sólo sería necesario modificar la CSS.

1.10. XSL

El W3C diseñó **XSL** en 1998 con el objetivo de promover especificaciones que ayudaran a realizar una web más común, rápida, accesible y de fácil mantenimiento.

Web recomendada

XSL (W3C). http://www.w3.org/Style/XSL/

XSL (*extensible style language*) es la especificación para separar el estilo del contenido cuando se crean páginas HTML o XML. Las especificaciones trabajan como plantillas, lo que permite a los diseñadores aplicar un único estilo de documento a múltiples páginas.

XSL es la segunda especificación de estilo ofrecida por el W3C. La primera fue la CSS, que es similar al XSL pero no posee dos de las mejoras del XSL:

1) Permitir a los desarrolladores el modo como las páginas web serán impresas.
2) Las especificaciones que permiten transferir documentos XML por diferentes aplicaciones.

1.11. XSLT

XSLT (*extensible stylesheet language transformations* o lenguaje de transformación basado en hojas de estilo extensible) es un lenguaje de programación, una aplicación XML que define cómo se transforma un documento XML.

Webs recomendadas

XSLT (W3C). http://www.w3.org/TR/xslt

J. Tomás Nogales Flores. "Introducción a XSLT (*Extensible Stylesheet Language - Transformations*)". http://www.bib.uc3m.es/~nogales/cursos/xslt.html

Normalmente, XSLT se utiliza para transformar los datos XML a formato HTML, o para convertirlo a otros formatos a fin de que el cliente lo pueda representar.

2. HTML y XHTML

2.1. Objetivos

Los objetivos de este apartado son conocer HTML y XML intensiva y extensivamente. En primer lugar, sus características principales y, en segundo lugar, las diferentes versiones que existen de ellos y la secuenciación y el sentido de éstas.

Por otra parte, el estudiante deberá conocer y saber identificar la estructura básica de un documento HTML y tener una idea general sobre las principales etiquetas que componen este lenguaje de marcado para la web.

2.2. Introducción

La publicación de información digital se puede llevar a cabo mediante diferentes formatos y medios. Una de las opciones que ofrece la Red es la edición web, que se fundamenta básicamente en los conceptos de hipertexto e hipermedia con los que podemos navegar por Internet.

Si entráis a cualquier página web con el navegador que tengáis definido por defecto, desde el menú **Visualiza**, elegid la opción **Fuente de la página** (**Ver | Código fuente de la página**). Se os abrirá una ventana con un código que, en una primera aproximación, puede parecer poco inteligible. Se trata del lenguaje **HTML** que los navegadores interpretan y nos presentan en forma de páginas web.

Código fuente HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html><head>
<meta content="text/html; charset=ISO-8859-1" http-equiv="content-type"><title>documento básico</title>

</head><body>
<br>
</body></html>
```

HTML corresponde al acrónimo de *hypertext mark language*, o lenguaje de marcado, diseñado para estructurar los documentos de textos y relacionarlos a modo de hipertexto en páginas web. Este lenguaje basa su sintaxis en un elemento de base que denominamos etiqueta (***tag***). Mediante las etiquetas se definen los elementos del documento, como enlaces, párrafos, imágenes, etc. Así, un documento HTML estará constituido por un texto y un conjunto de etiquetas para definir el modo mediante el cual se deberá presentar el texto y otros elementos en la página.

La sintaxis de HTML está estructurada según el protocolo:

1) Una etiqueta inicial que señala el comienzo de un elemento.

2) Un número determinado de atributos (y sus valores asociados).

3) Una cierta cantidad de contenido (caracteres y otros elementos).

4) Una etiqueta que marca el final.

En un principio, se pretendía que las etiquetas fueran capaces de marcar la información de acuerdo con su significado (títulos, enlaces, párrafos, listas, etc.), sin importar cómo se mostraba en la pantalla. Lo importante era el contenido y no la forma, es decir, estaba orientado a describir los contenidos. Se dejaba a cada visualizador (*browser*) la tarea de dar formato al documento según su criterio. Eso producía presentaciones diferentes, pero permitía controlar fácilmente el contenido.

Con el éxito de la web, la presión para conseguir controlar la forma en la que se presentaba la información llevó a que diferentes navegadores (Microsoft, Netscape) inventaran ampliaciones del HTML para conseguir la presentación deseada, y formatear y presentar texto y gráficos. El desbarajuste que eso provocó todavía lo sufrimos hoy, cuando para visualizar una página es necesario emplear un determinado navegador o cuando se deben crear diferentes versiones de la propia web para diferentes navegadores o dispositivos. Esto provocó la aparición de un consorcio que sirve de punto de encuentro con el fin de estudiar las necesidades futuras y proponer estándares: el **W3C (World Wide Web Consortium)**.

Estándares W3C

Misión del W3C

El Consorcio World Wide Web (W3C) es un consorcio internacional para desarrollar estándares web. La misión del W3C es: guiar la web hacia su máximo potencial mediante el desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la web (http://www.w3.org/).

2.3. Versiones de HTML

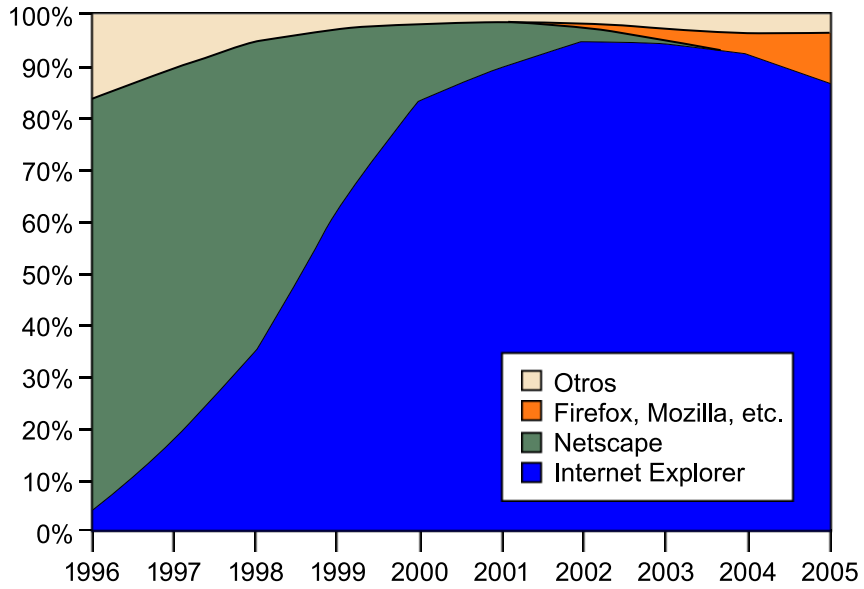
La historia completa de HTML es relativamente corta en el tiempo, pero densa en acontecimientos. El origen de HTML data del año 1980, cuando Tim Berners-Lee, físico del CERN (Organización Europea para la Investigación Nuclear), propuso un nuevo sistema de "hipertexto" para compartir documentos.

- **HTML 1.0 - 2.0** (1989-1991)

Fueron los primeros pasos del HTML; las páginas no eran muy atractivas pero contenían hipertexto (texto que enlaza a otra información), base de la web.

Así, a mediados de la década de 1990 empezaron las ampliaciones y evoluciones del HTML para mejorar la presentación de los documentos. HTML ha sido, pues, un lenguaje en constante evolución y a cada nueva versión se le ha dado un número. La primera versión definitiva fue **HTML 2.0**.

Guerra de los navegadores



- **HTML 3** (1995)

Éste fue el período de conflictos entre los navegadores (*Browser wars*[1]). Netscape y Microsoft competían por tener un navegador con más funciones (y así ganar mercado), hasta inventaban sus propias etiquetas HTML. En medio de esta guerra estaba el pobre desarrollador web, que debía estar al corriente de los dos navegadores.

(1)Guerra de los navegadores. http://es.wikipedia.org/wiki/Guerra_de_navegadores.

Ejemplo

Por ejemplo, Internet Explorer tenía la etiqueta **`<marquee>`** y Netscape la etiqueta **`<blink>`**.

En este período era común encontrar en las webs frases como "best seen on Internet Explorer".

HTML 3 surgió a finales de 1995 y fue un esfuerzo por mejorar las características y la utilidad de HTML. No obstante, nunca se llegó a completar.

- **HTML 3.2**

Fue la versión oficial con integración de soporte para tablas, imágenes, título y atributos de los elementos *align*, y algunos otros detalles minuciosos. HTML 3.2 es la versión que prácticamente todos los navegadores entienden.

- **HTML 4** (1998)

Fue el período en el que se empezaron a solucionar los conflictos entre los navegadores, gracias a la intervención del **Wold Wide Web Consortium (W3C)** y la creación de una sola versión de HTML.

La aportación principal del HTML4 fue separar la estructura y la presentación de las páginas web en dos lenguajes. HTML 4 para la estructura y CSS[2] para la presentación, y convencer a las compañías que creaban navegadores de que era necesario adoptar estos estándares.

(2) *Cascading style sheets.*

- **HTML 4.01** (1999)

HTML 4.01 es la versión más actual de HTML, y seguramente la más usada. Por fin se podía escribir tranquilamente un solo código estándar, seguro de que la mayoría de los navegadores lo interpretaría bien.

Con HTML 4.01 puedes estar seguro de que la gran mayoría de los navegadores mostrará el contenido de una web de la manera correcta.

Por lo tanto, HTML 4.01 es el estándar oficial vigente. Incluye soporte para la mayoría de las extensiones propietarias, además de soporte para características adicionales (internacionalizados de los documentos, el soporte para hojas de estilo: *cascading style sheets*, extra tabla, el formulario y mejoras de JavaScript) que no son universalmente compatibles.

No obstante, la evolución de HTML no ha cesado –HTML 4.01 es la última versión de HTML. En el futuro, el HTML será reemplazado por un nuevo idioma, denominado **XHTML** (*extensible hypertext markup language*). Las diferencias son realmente pequeñas, pero importantes, como veremos al hablar de él.

- **XHTML 1.0** (2000)

Las cosas cambiaron. HTML y otro lenguaje de marcado conocido como XML se juntaron y nació el XHTML 1.0.

Es un lenguaje que combina la popularidad y la capacidad de verse correctamente en todos los navegadores del HTML con la capacidad de extensión del XML.

Como vemos, la evolución del lenguaje HTML ha buscado la separación del contenido y la apariencia. Con la versión 4 del HTML se recomendaba otro mecanismo para controlar la visualización de nuestro contenido HTML: las hojas de estilo (CSS: *cascading style sheets*).

El W3C recomienda el uso del XHTML, que mantiene la misma sintaxis y los mismos mecanismos que el HTML, pero reformulado con las normas de un XML, con lo que se prepara así para aprovechar las ventajas de este lenguaje.

Por otro lado, el Web Hyperxtext Application Technology Group (**WHATWG**) –grupo de trabajo compuesto por la Fundación Mozilla y Opera Inc.– está planteando una especificación para un HTML 5, extendiendo el HTML 4.01 y el DOM. El HTML 5 intenta mejorar la parte de aplicación web con la especifi-

cación Web Forms 2.0. Este grupo surge como reacción al cambio brusco del paso de HTML a XHTML que, si no fuera por el Apéndice C de la especificación XHTML 1.0, no se podría usar en navegadores que no soportan el MIME type application/xhtml+xml.

Captura página principal WHATWG

No hay que entender el WHATWG como una organización paralela al W3C, sino como un grupo complementario, ya que cuando tiene un borrador lo propone al W3C para estandarizarlo.

La última especificación vigente es el XHTML 1.1, que ya no contempla ninguna compatibilidad con versiones anteriores y, por lo tanto, sólo se puede servir como application/xhtml+xml, excluyendo cualquier navegador antiguo.

El punto más polémico actualmente es la propuesta de especificación (en estado de borrador) XHTML 2.0, que deja de ser compatible con versiones anteriores, no sólo a nivel de MIME type, ya que la estructura de documento y los elementos estructurales cambian.

2.4. HTML 4. 01

El **HTML 4** desarrolló el lenguaje HTML con mecanismos para hojas de estilo, ejecución de *scripts*, marcos, objetos incluidos, soporte mejorado para texto de derecha a izquierda y direcciones mezcladas, tablas más ricas y mejoras en formularios y en la accesibilidad para personas con discapacidades. De hecho, **elHTML 4.01**[3] es una revisión de HTML 4.0, que corrige errores e introduce algunos cambios desde la revisión anterior.

(3)HTML 4.01 Specification W3C Recommendation 24 December 1999. http://www.w3.org/TR/html4.

1) Internacionalización

Permite que los documentos puedan ser escritos en cualquier idioma al adoptar el estándar *ISO/IEC: 10.646*, relacionado con la representación de caracteres internacionales (dirección del texto, puntuación y otros aspectos).

Hay también un mayor apoyo a los lenguajes humanos, lo que facilita la indexación de los documentos por parte de los motores de búsqueda, ofreciendo tipografía de más calidad, mejoras en la conversión de texto a voz, etc.

2) Accesibilidad

El crecimiento de la comunidad que utiliza la web ha provocado que la tecnología se adapte a finalidades específicas. El HTML se ha diseñado para hacer las páginas web más accesibles a aquellas personas que presentan limitaciones físicas. Los desarrollos de HTML 4 derivados de cuestiones de accesibilidad incluyen:

a) Distinción entre la estructura y la presentación de un documento. Se aconseja la utilización de hojas de estilo en lugar de elementos y atributos de presentación de HTML.

b) Mejores formularios, incluyendo la adición de teclas de acceso, la posibilidad de agrupar semánticamente los controles de un formulario, la posibilidad de agrupar las opciones **`SELECT`** semánticamente y los encabezados activos.

c) La posibilidad de codificar una descripción textual de un objeto incluido (con el elemento **`OBJECT`**).

d) Un nuevo mecanismo de mapas de imágenes en el lado del cliente (el elemento **`MAP`**) que permite a los autores integrar vínculos de imagen y de texto.

e) La exigencia de que se incluya texto alternativo acompañando las imágenes dentro del elemento **`IMG`** y en los mapas de imágenes dentro del elemento **`AREA`**.

f) Soporte de los atributos *title* y *lang* en todos los elementos.

g) Soporte de los elementos **`abbr`** y **`ACRONYM`**.

h) Un espectro más amplio de medios utilizables para hojas de estilo (***tty***, braille, etc.).

i) Tablas mejoradas, incluyendo títulos, grupos de columnas y mecanismos para facilitar su representación no visual.

j) Descripciones largas para tablas, imágenes, marcos, etc.

3) Tablas

Ahora hay más control sobre la estructura y la presentación (por ejemplo, grupos de columnas). La posibilidad de recomendar anchuras para las columnas permite a los navegador mostrar los datos de la tabla de manera incremental (a medida que los reciben), en lugar de tener que cargar toda la tabla antes de empezar a representarla.

Nota

En el momento en el que se escribió esta especificación, algunas herramientas de creación de HTML se basan exhaustivamente en el uso de tablas para dar formato a los documentos, lo que puede causar fácilmente problemas de accesibilidad.

4) Documentos compuestos

HTML ofrece ahora un mecanismo estándar para incluir objetos genéricos y aplicaciones dentro de documentos HTML. El elemento **`OBJECT`** (al lado de los antiguos elementos **`IMG`** y **`APPLET`**, más específicos) proporciona un mecanismo para incluir imágenes, vídeo, sonido, fórmulas matemáticas, aplicaciones especializadas y otros objetos en un documento. También permite especificar una jerarquía de representaciones alternativas para los navegadores que no soporten una representación específica.

5) Hojas de estilo

Las hojas de estilo simplifican el código HTML y liberan en gran medida el HTML de las responsabilidades de presentación. Ello permite control sobre la presentación de los documentos: fuentes, alineación, colores, etc.

La información de estilo se puede especificar para elementos individuales o para grupos de elementos, así como en un documento HTML o en hojas de estilo externas.

El mecanismo para asociar una hoja de estilo con un documento es independiente del lenguaje de la hoja de estilo.

Antes del uso de las hojas de estilo, los autores tenían un control limitado sobre la representación. **HTML 3.2** incluía un número de atributos y elementos que ofrecían control sobre la alineación, el tamaño de la fuente y el color del texto. Además, los autores utilizaban las tablas y las imágenes como medio de organizar la presentación de sus páginas. El tiempo relativamente largo que necesitan los usuarios para actualizar sus navegadores hará que estas características sigan siendo usadas durante algún tiempo. No obstante, al ofrecer las hojas de estilo mecanismos de presentación más potentes, el **World Wide Web Consortium** declarará obsoletos en el futuro muchos de los elementos y atributos de presentación del HTML. A lo largo de esta especificación estos elementos y atributos se marcan como "desaprobado".

6) Ejecución de *scripts*

Gracias a los *scripts*, los autores pueden crear páginas web dinámicas (por ejemplo, "formularios inteligentes", que reaccionan a medida que los usuarios los cumplimentan) y utilizar el HTML para crear aplicaciones en red.

Los mecanismos proporcionados para incluir *scripts* en un documento HTML son independientes del lenguaje de programación de los *scripts*.

7) Impresión

A veces, los autores querrán facilitar a los usuarios la impresión de su trabajo, sin limitarse al documento actual. Cuando un documento forme parte de un trabajo mayor, las relaciones entre los documentos se pueden describir mediante el elemento HTML LINK o utilizando el *resource description framework* (RDF, marco de descripción de recursos) del W3C.

2.5. Creación de documentos con HTML 4

Desde el W3C se recomienda a los autores e implementadores que sigan los siguientes principios generales cuando trabajen con HTML 4.

- **Separar estructura y presentación**

El HTML tiene sus raíces en **SGML**, lenguaje para la especificación de código estructural. A medida que el HTML madura, un número cada vez mayor de sus elementos y atributos presentacionales ha sido reemplazado por otros mecanismos, en particular las hojas de estilo. La experiencia ha demostrado que separando la estructura de un documento de sus aspectos presentacionales se reduce el coste de servir a un amplio espectro de plataformas, medios, etc., y se facilitan las revisiones del documento.

- **Considerar la accesibilidad universal en la web**

Para hacer la web más accesible a todos, especialmente a aquellas personas con discapacidades, los autores pueden considerar cómo quieren representar sus documentos en diferentes plataformas: navegadores basados en voz, lectores braille, etc. No estamos recomendando a los autores que limiten su creatividad, sólo que consideren representaciones alternativas de sus diseños. El HTML ofrecerá unas mecanismos con esta finalidad (por ejemplo, el atributo **`alt`**, el atributo **`accesskey`**, etc.)

Además de eso, los autores deberán recordar que sus documentos pueden llegar a una audiencia muy lejana con diferentes ordenadores y configuraciones. Para que los documentos sean correctamente interpretados, los autores deberían incluir en sus documentos información sobre el idioma natural y la dirección del texto, cómo está codificado el documento y otras cuestiones relacionadas con la internacionalización.

- **Ayudar a los agentes de usuario con la representación incremental**

Mediante un diseño minucioso de las tablas y haciendo uso de las nuevas características de las de HTML 4, los autores pueden ayudar a los agentes de usuario a representar los documentos más rápidamente. Los autores pueden aprender a diseñar tablas para su representación incremental (podéis consultar el elemento TABLE). Los implementadores deberían consultar las notas sobre tablas del apéndice para obtener información sobre algoritmos incrementales.

2.6. Estructura básica de un documento web

Es un lenguaje de marcado que deriva del SGML diseñado para estructurar textos y relacionarlos a modo de hipertexto. Desde el punto de vista físico, un documento HTML suele ser un fichero de texto ASCII (lo que lo hace transportable de una a otra plataforma), marcado con la sintaxis de lenguaje HTML, que puede ser interpretado por un *browser* o navegador.

Suele llevar la extensión de documento *".html"* (*".htm"* en DOS).

Se transmite por la Red como una secuencia de bytes con información de codificación, concretamente como un mensaje de tipo de contenido (*content-type*) "texto / html", con un parámetro opcional "charset" referido al sistema de codificación de caracteres.

- El documento está estructurado en elementos HTML anidados unos dentro de otros, lo que origina conceptos como elementos padres, ancestros, herencia de determinadas características o propiedades.

Un documento básico HTML 4 se compone de tres partes:

1) Una primera línea que contiene información sobre la versión del HTML; es la declaración que especifica el tipo documento, dentro de la etiqueta **<! DOCTYPE>**.

2) Una sección de cabecera declarativa, el elemento **<HEAD>** que se cierra con **</HEAD>**. Entre estas dos etiquetas se sitúa toda una serie de información relativa al documento: tipo de contenido, título, metadatos, enlaces, referencias, etc.

3) Un cuerpo, que contiene el contenido real del documento. El cuerpo puede ser especificado mediante el elemento **<BODY>** o mediante el elemento **<FRAMESET>** en los documentos que definen *frames*.

Declaración del tipo de documento

Declaración de tipo de documento (en SGML) e información de la versión HTML, situada al principio, referida a uno de los tres DTD de HTML 4.0 existentes, y que incluye el URI (habitualmente un URL) de la DTD, lo que facilita que el navegador lo pueda cargar si es necesario:

- **Estricto** (no contempla algunos elementos desaconsejados, relacionados sobre todo con la presentación física del documento, introducidos por anteriores versiones):

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

Transicional (contempla todos los elementos HTML, incluyendo los desaconsejados):

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

- **Con uso de *frames*** (como el transicional, incluyendo elementos relacionados con los *frames*):

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

Captura elementos básicos documento html

```
 1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
 2. <html>
 3. <head>
 4.   <meta content="text/html; charset=ISO-8859-1"
 5.  http-equiv="content-type">
 6.   <title>Documento básico en HTML 4.01</title>
 7. </head>
 8. <body>
 9. Elementos básics de un documento HTML4.01<br>
10. </body>
11. </html>
```

2.6.1. El elemento HTML

<HTML> ... **</HTML>**: principio y fin de documento.

- Elemento HTML, que engloba todo el resto del documento.

- Puede contener los **atributos de lengua o internacionalización** [4.0][4]:

(4) Lo que aparece cerrado entre corchetes es opcional, puede aparecer o no según determinadas situaciones.

": lang" = "lengua"

Ejemplos

<HTML lang="ca"> ...; **<DIV lang="es-ES">** ..., **<P lang="es-AR">** ..., **<Q lang = "a">**..., **<SPAN lang="fr">** ...

dir = "(ltr | rtl)": [4.0] Indica la dirección del texto neutral (no determinado por Unicode), de izquierda a derecha (*ltr*) o viceversa (*rtl*). Puede acompañar *lang* en cualquier elemento en el que éste figure.

El elemento **<HTML>** debe incluir dos elementos obligatorios: **<HEAD>** y **<BODY>**.

El cuerpo incluye el contenido visible del documento en el navegador. Se inicia con la etiqueta **<BODY>** y se cierra con la etiqueta **</BODY>**. Cualquier elemento, carácter, etc., del contenido que no se encuentre entre las dos etiquetas **<BODY>** ... **</BODY>** invalidará el documento.

Pueden aparecer blancos (espacios, saltos de línea, tabulaciones y comentarios) antes y después de cada sección. Las secciones 2 y 3 deben estar delimitadas por el elemento HTML que se inicia con **<HTML>** y se cierra con **</HTML>**.

2.7. Etiquetas básicas

2.7.1. Concepto de etiqueta

La etiqueta presenta frecuentemente dos partes:

Una apertura de forma general <etiqueta>

Un cierre de tipo </etiqueta>

Todo el texto incluido en el interior de esta etiqueta sufrirá las modificaciones que caracterizan a esta etiqueta. Así, por ejemplo:

Las etiquetas **** y **** definen un texto en negrita. Si en nuestro documento HTML escribimos una frase con el siguiente código:

****Eso está en negrita****

El resultado será:

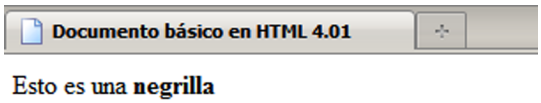
Eso está en negrita

En HTML básico el código sería:

Código HTML de un texto con negrita

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2. <html>
3. <head>
4.   <meta content="text/html; charset=ISO-8859-1"
5.  http-equiv="content-type">
6.   <title>Documento básico en HTML 4.01</title>
7. </head>
8. <body>
9. Esto es una <b>negrilla</b>
10. <br>
11. </body>
12. </html>
```

Visualización en el navegador



2.7.2. Cabecera de un documento: <HEAD>...</HEAD>

La cabecera contiene toda la información sobre el documento: el título, las etiquetas META, BASE, hojas de estilo, *scripts*, etc.

<HEAD>... **</HEAD>**: indica principio y fin de la sección del encabezamiento. Aquí irán etiquetas que definirán el cabezal o toda la página.

La cabecera de un documento **HTML**, tal como ya hemos comentado, se abre con la etiqueta **<HEAD>** y se cierra con **</HEAD>**.

Contiene datos de identificación y especificaciones del documento.

Código fuente elemento HEAD

```
 1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
 2. <html style="direction: ltr;" lang="es-es">
 3. <head>
 4.   <meta content="text/html; charset=ISO-8859-1"
 5.  http-equiv="content-type">
 6.   <title>Documento básico en HTML 4.01</title>
 7.   <meta content="UOC" name="author">
 8.   <meta content="Lenguaje HTML" name="description">
 9. </head>
10. <body>
11. Esto es una <b>negrilla</b>
12. <br>
13. </body>
14. </html>
```

Además de los atributos de internacionalización (*lang* y *dir*), puede llevar otros atributos opcionales como:

profile = *"URL"*: [4.0] Especifica la URL de un perfil de metadatos (por ejemplo, el *Dublin Core*, que fija las propiedades recomendadas para descripciones bibliográficas electrónicas), que afectaría a todos los elementos **<meta>** y **<LINK>** de **<HEAD>**. El formato de los perfiles no está fijado en la especificación HTML 4.0.

El único elemento que debe aparecer obligatoriamente en la cabecera es el título del documento, que se abre con la etiqueta **<TITLE>** y se cierra con **</TITLE>**. También es requerido, aunque no imprescindible, la declaración del tipo de contenido y el juego de caracteres.

2.7.3. Cuerpo del documento: <BODY>...</BODY>

En el cuerpo del documento es donde se incluye y da formato al contenido que se quiere mostrar en la página web: texto, imágenes, gráficos, enlaces a otras páginas, listas, formularios, etc.

<BODY>...</BODY> Indica principio y fin del cuerpo del documento. Todo lo que queremos que se visualice irá dentro de esta sección. Es el contenido real del documento, que se mostrará en la ventana del navegador.

<FRAMESET>

Un documento que define la división de la ventana en *frames* y establece el contenido inicial de cada *frame* no tiene elemento **<BODY>**, sino **<FRAMESET>**.

El **<BODY>** puede llevar una serie de atributos opcionales:

1) Atributos del núcleo (*core attributes*)

id = "identificador": Fija un identificador, con un nombre único a lo largo del documento, que se asocia al elemento en cuestión y que después puede referenciar desde el mismo a otro documento.

class = "clase [clase]...": [4.0] Fija una o más clases o categorías (separadas por espacios) que se asocian al elemento y pueden referenciar desde una hoja de estilo. Los atributos ***id*** y ***class*** son identificadores de elemento.

style = "estilo": [4.0] fija una serie de características de estilo para el elemento, expresadas según un determinado lenguaje de estilo por defecto (como CSS).

title = "texto": [4.0] Establece un texto como título para el elemento, que el navegador mostrará de alguna manera en ciertas situaciones (por ejemplo, en un gráfico, mostrando una etiqueta con el texto cuando el puntero se sitúa sobre el elemento).

Pueden también asociarse a la mayor parte de los elementos incluidos en **<BODY>**.

2) Atributos de acontecimientos intrínsecos (*intrinsic events*)

Toman la forma *acontecimiento* = *"script"* [4.0] y determinan que se ejecute el *script* si se produce el acontecimiento.

3) Atributos de presentación o estilo del documento

Todos están desaconsejados en la especificación HTML 4.0; en su lugar se deberían establecer las oportunas indicaciones en una hoja de estilo.

background = *"URL"*: Especifica, por medio de un URL, absoluto o relativo, un fichero de imagen con el que se llena el fondo.

(*bgcolor* | *text* | *link* | *VLINK* | *alink*) = "color": Fijan un color para el fondo (*bgcolor*), el texto (*text*) y los enlaces no visitados (*link*), visitados (*VLINK*) y activos (*alink*).

El color que asignan estos atributos (y otros) puede especificar de dos maneras:

Código color

Black #000000	Green #008000	Silver #C0C0C0	Lime #00FF00
Gray #808080	Olive #808000	White #FFFFFF	Yellow #FFFF00
Maroon #800000	Navy #000080	Red #FF0000	Blue #0000FF
Purple #800080	Teal #008080	Fuchsia #FF00FF	Aqua #00FFFF

por nombre (***black, white, red, yellow, gray***, **etc.**), o

por valor RGB, en la forma *# rrggbb*, ex.: ***# 000000, # FFFFFF, # FF0000, # FFFF00, # 808.080...***)

4) Comentarios

`<! - Comentari ->`: se utiliza esta etiqueta para incluir comentarios, anotaciones, dentro del código fuente del documento. Las anotaciones siempre son útiles cuando se quieren hacer modificaciones. También se utiliza para esconder todo aquel código fuente que no se quiere mostrar en el programa navegador sin necesidad de borrarlo. Todo lo que se encuentre entre **`<!-- y -->`** será ignorado por el programa navegador.

Inserta en cualquier lugar del elemento **`<HTML>`** un comentario que es ignorado por el navegador y que es útil para la depuración o documentación de la página HTML.

El formato es general por ***SGML***. "**`<!": Comienzo declaración de marcado; "-": comienzo comentario; "-": ningún comentario; ">`**": ninguna declaración de marcado, no puede haber espacios dentro del grupo de caracteres "**`<! -", ni incluir el grupo "-" dentro del comentario`**.

Ejemplo

```
<! - Enlaces a revisar ->
```

2.7.4. Elementos propios de <HEAD>

1) El elemento <TITLE>...</TITLE>

El título del documento que aparece en la ventana del navegador se escribe entre estas dos etiquetas. Hay que tener en cuenta que algunos buscadores utilizan el título para indexar la página web en sus bases de datos. Es importante poner título en todas las páginas (incluso aunque se utilicen marcos).

<TITLE>Título de documento**</TITLE>**

Elemento obligatorio y único que especifica el título del documento (que debería ser bastante representativo), que suele mostrarse en la barra de la ventana del navegador para asociarse a los *bookmarks*, al historial de la navegación, etc. No puede incluir otros elementos, pero sí entidades de carácter. Puede llevar los atributos de internacionalización (*lang* y *dir*).

Hay otros elementos opcionales que pueden aparecer en la cabecera, como son el elemento:

<BASE> Elemento vacío y opcional que fija un **URL absoluto** (no puede ser una ruta) que se utilizará como base para resolver **URL relativos**: en caso de no existir, se utilizará como base la URL del documento; si existe, debe aparecer antes de cualquier elemento que contenga una referencia externa.

<META><META {name | Http-equiv}=" propiedad "content=" valor" [scheme=" esquema "]> ... Elemento vacío que permite crear una serie de metadatos o comunicar con el servidor HTTP. Debe llevar necesariamente uno de los atributos *name* o http-equiv (y puede llevar los atributos de lengua (*lang* y *dir*).

Con *name* describe propiedades del documento (datos sobre el documento o metadatos) en forma de pares **propiedad/valor**, estando de acuerdo el valor expresado con la convención **[4.0] esquema**.

Propiedades típicas son *autor, copyright, date, keywords, description, organization, lang*, etc.

Los metadatos no se muestran directamente, pero deben ser accesibles al usuario ya que son muy útiles para facilitar la indexación de documentos por robots de busca.

2) El elemento <LINK>

Para vincular el documento de HTML con el fichero externo que contiene definidos los estilos se incluye esta etiqueta dentro de la cabecera del documento. Las hojas de estilo se pueden guardar separadas del documento de HTML en un fichero de texto sin formato (*nom_fichero*.css) externo.

```
<LINK href = "URL" rel = "relación" [rev = "relación"]>
```

Elemento vacío que define **relaciones** típicas del documento con otros documentos HTML o recursos web. Pensado inicialmente para barras de navegación genéricas o para la impresión conjunta de diferentes documentos relacionados, ha sido poco usado y algunos navegadores lo ignoran; sin embargo, la información que aportan podría ser muy útil para los robots de busca.

Los enlaces indicados no se muestran en el contenido del documento.

Puede llevar atributos de núcleo, de acontecimientos básicos y de lengua, y además:

href = "URL": Establece el URL completo (o bien la *ruta*, o bien URL incompleto) del documento con el que está relacionado.

type = "tipo_contenido": Fija el tipo de contenido MIME del recurso referenciado en href. Debe ser una entrada del registro de la IANA. Por ejemplo: "texto / html", "texto / css", "image / jpg", "video / mpeg", etc.

charset = "sistema_codificación": Define la codificación de caracteres utilizada en el recurso referenciado en href. Por ejemplo: "ISO-8859-1" (valor por defecto) o "EUC-JP".

hreflang = "lengua": Anuncia la lengua base del documento referenciado en href.

media = "medio [, medio]...": Fija los tipos de medios, dispositivos o soportes, para los que está diseñado el recurso referenciado en href. Puede usarse cuando el atributo raíz toma los valores *stylesheet* o *alternate*.

raíz = "relación": Fija el tipo de relación o enlace existente entre el documento actual y el documento referenciado en href. La relación puede ser una de las siguientes:

alternate: Establece un enlace a una versión alternativa del documento actual, en otra lengua u otro soporte o formato. Por ejemplo:

Ejemplo

```
<LINK title = "Manual en Portugués" type = "text / html" rel = "alterna-
te" hreflang = "pt" href = "http://ord.filial.pt/manual/portug.html">
```

```
<LINK media="print" title="Manual en Postscript" " type = "applica-
tion / postscript" rel = "alternate" href="http://ord.empresa.com/
manual/postscript.ps">
```

{*start* | *next* | *prev*}: Señalan respectivamente los documentos inicial, siguiente y anterior en una colección ordenada, en relación con el documento actual.

{*contents* | *index* | *glossary*}: Señalan documentos que actúan con sumario, índice o glosario para el documento actual.

copyright: Señala un documento que contienen los datos de propiedad intelectual del documento actual.

```
<head>
<title>Capítulo 3</title>
<LINK rel="start" href="../intro.html">
<LINK rel="contents" href="../toc.html">
<LINK rel="index" href="../indice.html">
<LINK rel="glossary" href="../glosario.html">
<LINK rel="copyright" href="../copy.html"> <LINK Rel="next" href="capit4.html">
<LINK Rel="prev" href="capit2.html">
</HEAD>
```

Código elemento link

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2. <html style="direction: ltr;" lang="es-es">
3. <head>
4.   <meta content="text/html; charset=ISO-8859-1"
5.  http-equiv="content-type">
6.   <title>Documento básico en HTML 4.01</title>
7.   <meta content="UOC" name="author">
8.   <meta content="Lenguaje HTML" name="description">
9.   <link rel="start" href="../intro.html">
10.  <link rel="contents" href="../toc.html">
11.  <link rel="index" href="../indice.html">
12.  <link rel="glossary" href="../glosario.html">
13.  <link rel="copyright" href="../copy.html">
14.  <link rel="next" href="capit4.html">
15.  <link rel="prev" href="capit2.html">
16. </head>
17. <body>
18. <br>
19. </body>
20. </html>
```

{*chapter* | *section* | *subsection* | *appendix* | *help*}: Señalan documentos que actúan como capítulo, sección, subsección, apéndice o ayuda en una colección de documentos.

bookmark: Señala un punto de entrada clave en un documento extenso. Puede haber varios en el mismo documento, y podrían etiquetar con el atributo title.

stylesheet: Especifica para el documento una hoja de estilo en un fichero separado; la localización y el lenguaje de estilo se expresan respectivamente con *href = "URL"* y *type = "tipoContenido"*.

rel="stylesheet ", **<LINK>** puede llevar también el atributo medio = "medio [, medio]..." para señalar los medios o dispositivos previstos en la hoja de estilo (o con rel = "alternate", para indicar otros soportes en los que puede encontrarse el mismo documento): *[screen]* | *tv* | *tty* | *projection* | *hanheld* | *print* | *braille* | *aural* | *all P. P*.

> Por ejemplo, **<LINK rel="stylesheet" href="estilos/normal.css" type="text/css">** establece para el documento la hoja de estilo "normal.css", situada en el subdirectorio "estilos" (con relación al documento HTML) y escrita en el lenguaje de estilo CSS.

.rev = "relación [relación]...": Fija las relaciones inversas de raíz, es decir, el tipo de relación que une al documento enlazado mediante href con el documento actual. Si un documento "doc1" contiene **<LINK href="doc2" rel="contents">**, "doc2" podría contener **<LINK href="doc1" rev="contents">**. Si un documento "doc1" contiene **<LINK href="doc2" rel="next" rev="prev">**, "doc2" podría contener **<LINK href="doc1" rel="prev" rev="next">**.

Puede llevar también el atributo *target*= "*frame_destino*", tratado en relación con el elemento **<A herf="URL">**.

3) El elemento <STYLE>

Se pueden definir los estilos necesarios para un documento insertándolos dentro de la cabecera entre las etiquetas **<STYLE>** y **</STYLE>**.

<STYLE type=" lenguaje ">especificaciones_de_estilo**</STYLE>**

- [4.0] Permite incluir dentro del mismo documento HTML especificaciones de estilo, escritas en un determinado lenguaje de hojas de estilo.

2.7.5. Elementos de bloque: secciones de texto

1) El elemento <P>

Un párrafo es un bloque de texto, delimitado por la etiqueta de apertura y por la de cierre, que se muestra separado del resto por un salto de línea.

Código elemento párrafo

```
17. <body>
18. <br>
19. <p>"Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
20. do
21. eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
22. minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip
23. ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
24. voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
25. sint occaecat cupidatat non proident, sunt in culpa qui officia
26. deserunt mollit anim id est laborum."
27. </p>
28. </body>
29. </html>
```

<P> ... </P>

- Indica el comienzo de un párrafo.
- Puede contener otros elementos de bloque.

Puede llevar atributos de núcleo, de acontecimientos básicos y de lengua y atributo align (desaconsejado).

Nota

Cuando se procesa el documento HTML, uno o más retornos de carro o espacios consecutivos se interpretan como un espacio.

Ejemplo

<P> Primer párrafo **<P>** Segundo párrafo

2) Elementos que agrupan elementos de bloque o de texto: <DIV> y <SPAN>

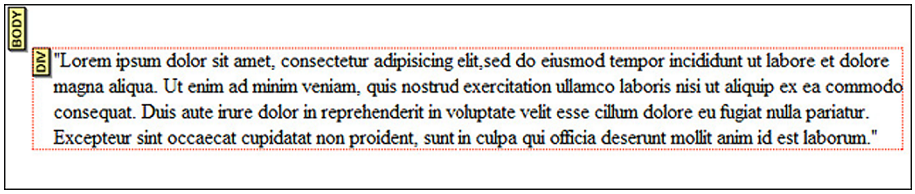
Se utiliza la etiqueta DIV para agrupar elementos con el mismo estilo. Siempre hay un salto de línea antes y después de los elementos agrupados por esta etiqueta

<DIV>Elementos_de_bloque**</DIV>**

Código elemento DIV

```
17. <body>
18. <br>
19. <div>"Lorem ipsum dolor sit amet, consectetur adipisicing elit,sed do
20. eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
21. minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip
22. ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
23. voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
24. sint occaecat cupidatat non proident, sunt in culpa qui officia
25. deserunt mollit anim id est laborum."
26. </div>
27. </body>
28. </html>
```

Vista desde el editor web en modo normal del elemento DIV



La etiqueta SPAN también sirve para agrupar elementos que deben tener el mismo estilo pero sin forzar un salto de línea antes y después de los elementos agrupados

<SPAN>Elementos_de_texto**</SPAN>**[4.0]

- Sirven para agrupar respectivamente elementos de bloque y texto.

La agrupación de elementos se realiza generalmente para asignarles una serie de propiedades comunes, directamente o referenciando un valor de su atributo *class* (o el valor de su atributo *id*) por medio de las hojas de estilo.

- Pueden llevar como atributos los del núcleo (especialmente los identificadores de elemento citados), de eventos básicos y de lengua.

Código fuente elemento SPAN

```
19. <span style="font-style: italic;">"Lorem ipsum
20. dolor sit amet, consectetur adipisicing
21. elit,sed do
22. eiusnod tempor incididunt ut labore et dolore magna aliqua. <span
23.  style="font-weight: bold;">Ut enim ad
24. minin veniam, quis nostrud exercitation</span> ullamco laboris
25. nisi ut aliquip
26. ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
27. voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
28. sint occaecat cupidatat non proident, sunt in culpa qui officia
29. deserunt mollit anim id est laborum."
30. </span>
```

Vista desde el editor web en modo normal del elemento SPAN

BODY

SPAN *"Lorem ipsum dolor sit amet, consectetur adipisicing elit,sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.* SPAN ***Ut enim ad minim veniam, quis nostrud exercitation*** *ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."*

<DIV> puede llevar el atributo opcional: align={[left] | right | center | justify} que fija la alineación horizontal de los elementos agrupados. Desaconsejado a favor de las hojas de estilo.

Nota

El valor *"justify"* no es soportado por muchos navegadores.

3) Elementos de bloque: Listas

Se utilizan para representar elementos en forma de lista. Existen tres tipos de listas: las no ordenadas, las ordenadas y las listas de definiciones; también se puede insertar una lista dentro de otra.

a) Tipo de listas: elementos **<OL>**, **<UL>**

<OL>lista**</OL>**: Lista ordenada (o numerada) (*ordered list*).

- Cada ítem de la lista se numera (su posición es significativa).

- Atributos opcionales (los dos desaconsejados en favor de las hojas de estilo):

type= {[1] | A | a | I | i}: Determina el tipo de ordenación: cifras arábigas, orden alfabético o números romanos[5].

(5)En mayúsculas o minúsculas.

start= n: Da el valor *n* al primer ítem[6] de la lista.

(6)Por defecto: 1.

Código fuente lista numerada

```
19. <ol>
20.   <li>"Lorem ipsum dolor sit amet, consectetur adipisicing
21. elit,sed </li>
22.   <li>do eiusmod tempor incididunt ut labore et dolore magna aliqua. </li>
23.   <li>Ut enim ad
24. minim veniam, quis nostrud exercitation ullamco</li>
25.   <li> laboris
26. nisi ut aliquip
27. ex ea commodo consequat. </li>
28.   <li>Duis aute irure dolor in reprehenderit in
29. voluptate velit esse </li>
30.   <li>cillum dolore eu fugiat nulla pariatur. </li>
31.   <li>Excepteur
32. sint occaecat cupidatat non proident, </li>
33.   <li>sunt in culpa qui officia
34. deserunt mollit anim id est laborum."
35.   </li>
36. </ol>
```

Visualización de lista numerada

1. "Lorem ipsum dolor sit amet, consectetur adipisicing elit,sed
2. do eiusmod tempor incididunt ut labore et dolore magna aliqua.
3. Ut enim ad minim veniam, quis nostrud exercitation ullamco
4. laboris nisi ut aliquip ex ea commodo consequat.
5. Duis aute irure dolor in reprehenderit in voluptate velit esse
6. cillum dolore eu fugiat nulla pariatur.
7. Excepteur sint occaecat cupidatat non proident,
8. sunt in culpa qui officia deserunt mollit anim id est laborum."

<UL>lista**</UL>**: Lista no ordenada (*unordered list*).

- Cada ítem se precede de una marca (por defecto cambia según el nivel de anidamiento).

Atributo opcional (desaconsejado): *type*={*disc* | *square* | *circle*}: marca que precede a cada ítem.

Código fuente lista no numerada

```
19. <ul>
20.   <li>"Lorem ipsum dolor sit amet, consectetur adipisicing
21. elit,sed </li>
22.   <li>do eiusmod tempor incididunt ut labore et dolore magna
23. aliqua. </li>
24.   <li>Ut enim ad
25. minim veniam, quis nostrud exercitation ullamco</li>
26.   <li> laboris
27. nisi ut aliquip
28. ex ea commodo consequat. </li>
29.   <li>Duis aute irure dolor in reprehenderit in
30. voluptate velit esse </li>
31.   <li>cillum dolore eu fugiat nulla pariatur. </li>
32.   <li>Excepteur
33. sint occaecat cupidatat non proident, </li>
34.   <li>sunt in culpa qui officia
35. deserunt mollit anim id est laborum."
36.   </li>
37. </ul>
```

Visualización lista no numerada

- "Lorem ipsum dolor sit amet, consectetur adipisicing elit,sed
- do eiusmod tempor incididunt ut labore et dolore magna aliqua.
- Ut enim ad minim veniam, quis nostrud exercitation ullamco
- laboris nisi ut aliquip ex ea commodo consequat.
- Duis aute irure dolor in reprehenderit in voluptate velit esse
- cillum dolore eu fugiat nulla pariatur.
- Excepteur sint occaecat cupidatat non proident,
- sunt in culpa qui officia deserunt mollit anim id est laborum."

b) Ítems de listas: `<LI>`

`<LI>`íDocument**`</LI>`**: Ítems de listas **`<OL>`**, **`<UL>`**

- Cada ítem empieza una nueva línea.

En una **`<OL>`**, **`<LI>`** puede llevar el atributo desaconsejado value = n para alterar la numeración de la lista a partir de este documento, que pasa a ser n.

- El término se muestra en una línea y la descripción, sangrada, en la misma línea si es posible y se utiliza el atributo compact (desaconsejado), o en las siguientes.

2.7.6. Elementos de texto: estilos/aspectos

Si se opta por el denominado ***marcado de presentación*** para indicar cada atributo del texto (tamaño, fuente, color, etc.), la edición será más lenta y la actualización futura será una tarea pesada y dificultosa.

Se recomienda identificar con los atributos específicos previamente cada uno de los elementos relevantes del texto y dejar que sean controlados por el navegador o por hojas de estilos en cascada. Es el llamado ***marcado estructural***. Este método de trabajo resulta muy flexible y más accesible.

Las etiquetas siguientes marcan (definen) cuál es el estilo del texto que hay entre la etiqueta de apertura y la de cierre.

1) Físicos (tipográficos)

- Atribuyen características exclusivamente tipográficas o de visualización al texto.
- Aunque algunos no están formalmente desaconsejados, se recomienda utilizar en su lugar especificaciones de hojas de estilo.
- Pueden llevar atributos de núcleo, de acontecimientos básicos y de lengua.

a) Elementos de estilo de fuente: `<B>`, `<I>`, `<BIG>`, `<SMALL>`, `<U>`

`<B>`Texto**`</B>`**: Texto en negrita (*bold*)

`<I>`Texto**`</I>`**: Texto en cursiva (*italics*)

`<BIG>`Texto**`</BIG>`**: Texto en tamaño grande (equivale a **`<FONT size=+1>`**Texto**`</FONT>`**)

`<SMALL>`Texto**`</SMALL>`**: Texto en tamaño pequeño (equivale a **`<FONT size=-1>`**Texto**`</FONT>`**)

`<U>`Texto**`</U>`**: Texto subrayado (a veces suele ignorarse, para no confundir con los vínculos). Desaconsejado.

Código fuente elementos de estilo de fuente

```
18. <b> Texto</b>: Texto en negrita (bold)<br>
19. <i>Texto</i>: Texto en cursiva (itálica) <big><br>
20. Texto</big>: Texto en medida grande (equivale a <font
21.  size="+1">
22. texto</font> ) <small><br>
23. Texto</small>: Texto en medida pequeña (equivale a <font
24.  size="-1">
25. texto</font>)<br>
26. <u>Texto</u>: Texto subrallado (a veces suele ignorarse,
27. para no confundirlo con los vínculos). Desaconsejado.
```

Visualización de elementos de estilo de fuente

Texto: Texto en negrita (bold)
Texto: Texto en cursiva (itálica)
Texto: Texto en medida grande (equivale a texto)
Texto: Texto en medida pequeña (equivale a texto)
Texto: Texto subrallado (a veces suele ignorarse, para no confundirlo con los vinculos). Desaconsejado.

2) Lógicos (semánticos): elementos de frase

Los elementos **<EM>**, **<STRONG>**

<EM>Texto**</EM>**: Texto enfatizado (*emphasis*)

<STRONG>Texto**</STRONG>**: Texto con fuerte énfasis.

Código fuente elementos *em* y *strong*

```
18. <em> Texto </em>: Texto emfatizado (emphasis) <br>
19. <strong>Texto</strong>: Texto con emfasi fuerte
```

Visualización de elementos *em* y *strong*

Texto : Texto emfatizado (emphasis)
Texto: Texto con emfasi fuerte

3) Vínculo (elementos de enlaces hipertextuales): el elemento **<A>**

La directiva para establecer un vínculo o un vínculo a un archivo es la siguiente:

<A {name="Identificador"| href="URL"}>vinculo**</A>**

Código fuente elementos A

```
20. "Lorem ipsum dolor sit amet, <a href="http://www.uoc.edu">consectetur</a>
21. adipisicing elit, sed do
```

Visualización de elemento A

"Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Hay que denominar con precisión cada uno de los componentes de esta estructura

- Permite establecer el vínculo de origen o de destino de un enlace hipertextual.

La capacidad para expresar una conexión (enlace, enlace web, etc.) entre un documento HTML y otro recurso web es lo que caracteriza al HTML frente a otros lenguajes de marcado; estos enlaces se expresan fundamentalmente por medio del elemento **<A>**

- Puede llevar atributos de núcleo de acontecimientos básicos y de lengua.

Toma dos formas según establezca un vínculo de origen o de destino (ya que en la Red los enlaces son unidireccionales), con los atributos *name* y *href*, aunque un mismo elemento **<A>** puede llevar dos atributos, de manera que un vínculo de partida puede ser utilizado también como vínculo de destino.

4) Vínculos de destino: el elemento **<A>** con el atributo name

<A Name="identificador">[vinculo_de_destino]**</A>**

Establece un nombre identificador (identificador de fragmento) que permitirá hacer futuras referencias hipertextuales al punto del documento donde se sitúa este elemento (vínculo de destino).

<H2><A name="ap3">Apartado 3 **</A> </H2>**

Cualquier elemento con un atributo id (incluso **<A id=" identificador ">**) puede ser también considerado y utilizado como un vínculo de destino.

<H2 id="ap3">Apartado 3**</H2>**

El identificador fijado con el atributo name

- Debe estar formado sólo por caracteres ASCII (aunque a diferencia del atributo id, puede contener referencias de carácter). Ser único a lo largo del documento (contando también los valores que toman atributos id usados, que comparten el mismo espacio de nombres), no pudiendo diferir sólo en la caja de las letras (mayúsculas o minúsculas).
- Pero a la hora de referenciar en un enlace sí se debe respetar la caja de las letras.

El elemento puede estar vacío[7] (en cualquier caso, su contenido nunca se destaca), pero requiere la etiqueta final.

(7)Algunos navegadores no reconocen vínculos vacíos.

<A name="ap3"> </A> <H2> Apartado 3 **</H2>**

5) Vínculos de origen: el elemento **<A>** con el atributo href

<A Href="{ URL [# identificador] | [URL]# identificador }"> vinculo**</A>**

Especifica un vínculo de origen o de partida (que el navegador resalta de alguna manera, por ejemplo con subrayado y/o diferente color, para que el usuario pueda identificarlo) y el correspondiente hiperenlace por medio de un URL, absoluto o relativo, a un recurso.

Cuando el recurso referenciado es un documento HTML, href puede incluir un identificador de fragmento, de manera que el enlace no se produce al principio del documento, sino en el punto concreto en el que se haya fijado este identificador (por medio de **<a name = "ident">** o algún elemento con el atributo id = "ident"). El identificador de fragmento puede constituir por sí mismo el valor de href; en este caso se entiende que pertenece al propio documento.

También permite especificar enlaces a otros recursos que no sean páginas HTML. El vínculo puede tener anidados elementos de texto en línea (por ejemplo, un elemento img pero no un elemento **<A>**.

Puede llevar los atributos charset, type, hreflang, rel y re, tratados a propósito en el elemento LINK y, cuando se usa dentro de un elemento **<map>** también shape y coords, tratados en relación con el elemento **<AREA>**.

a) Atributos de enfoque opcionales de <A href="URL"> y otros elementos:

accesskey = "carácter":

Asigna una tecla de acceso (un único carácter) a un elemento, que es enfocado cuando el usuario la aprieta (normalmente con otra, como *alt* o *command*), el navegador debe mostrar convenientemente la tecla de acceso.

El efecto de recibir el enfoque depende del elemento que lleve este atributo: si se trata de un elemento **<A>** o **<AREA>**, se activa el enlace, en formularios; si es un botón de radio o cuadro, cambia de *on* a *off* o viceversa; si se trata de un campo para introducir texto, se activa, etc.

tabindex = "n":

Asigna al elemento una posición *n* (0... 32767) en el orden de tabulación cuando se navega por el documento por medio del teclado (tabulador), de manera que se seleccionan consecutivamente los diferentes elementos. Para activar un elemento seleccionado, se requiere generalmente la pulsación de otra tecla (habitualmente el retorno).

El orden que se sigue navegando por tabulador es el siguiente: elementos con *tabindex* = n> 0, con vista desde el menor *n* al mayor (si hay valores repetidos, en orden de aparición); después, elementos sin *tabindex* o con *tabindex = 0*, en orden de aparición, los elementos inhabilitados (sólo relacionados con formularios) no participan.

onfocus = "script": Se ejecuta el *script* si el elemento es enfocado por el puntero o en la navegación por tabuladores. Un elemento enfocado se selecciona.

onblur = "script": Se ejecuta el *script* si el elemento deja de ser enfocado.

b) Otros atributos

target = "frame_destino":

No definido en el DTD estricto, sino en el DTD de documentos frameset. Especifica un *frame de destino* para el recurso referenciado en href, en el caso de que éste se active.

Además de los nombres que puede asignar el usuario a los *frames* que define (por medio del atributo name del elemento **<frame>** y que deben empezar por una letra), hay nombres reservados para designar una nueva ventana ("_blank"), el *frame* actual ("_self"), el *frame* padre ("_parent") o la ventana actual completa ("_top").

Este atributo puede llevar, además de **<A>**, los elementos **<LINK>**, **<AREA>** y **<form>**.

En un entorno de *frames*, el documento o recurso referenciado por todos estos elementos se carga, si llevan el atributo target, en el *frame* especificado por éste y, si no lo llevan, en el especificado por el atributo target del elemento **<BASE>**; en otro caso, se cargan en el mismo *frame* en el que estaba situado el documento. Si se especifica con target un *frame* de nombre desconocido, se abrirá una nueva ventana con un único *frame*, al que se le asigna este nombre y se carga en él el recurso.

El atributo title, opcional, en versiones anteriores de HTML se usaba sólo con el elemento **<A href="...">**, con el fin de dar un título a documentos sin título explícito (como menús de gopher o directorios de FTP), o un título provisional a un documento HTML, hasta que se cargara su elemento **<TITLE>**.

2.7.7. Elementos bloque: tablas

1) El elemento <TABLE>

HTML 4.0 ha modificado sustancialmente el modelo de tabla (pero manteniendo la compatibilidad hacia atrás), sobre todo con vistas a que pueda ser representada a medida que el navegador la recibe, sin la necesidad de esperar como hasta ahora a recibir la tabla completa. Por ello, la tabla debe contener al principio datos sobre el número de columnas y la anchura de éstas.

Puede llevar atributos de núcleo, de sucesos básicos y de lengua (el atributo dir establece, además de la direccionalidad del texto, la de la tabla, es decir, si la primera columna de una fila está a la izquierda o a la derecha). Estos atributos pueden llevar también todos los elementos propios de **<TABLE>**.

Elementos de table, tbody, tr, td

```
19. <table>
20.   <tbody>
21.     <tr>
22.       <td>
23. "Lorem ipsum dolor sit amet, consectetur
24. adipisicing elit, sed do
25. eiusmod tempor incididunt ut labore et dolore magna aliqua.
26.       </td>
27.       <td>Ut enim ad minim veniam, quis nostrud exercitation
28. ullamco laboris nisi ut aliquip
29. ex ea commodo consequat.
30.       </td>
31.     </tr>
32.     <tr>
33.       <td>
34. Duis aute irure dolor in reprehenderit in
35. voluptate velit esse cillum dolore eu fugiat nulla pariatur.</td>
36.       <td> Excepteur
37. sint occaecat cupidatat non proident, sunt in culpa qui officia
38. deserunt mollit anim id est laborum."
39.       </td>
40.     </tr>
41.   </tbody>
42. </table>
```

Visualización en el editor web elementos de table, tbody, tr, td

"Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.	Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

<TABLE>Elementos_de_Tabla**</TABLE>**

Las celdas pueden tener cualquier contenido (texto, imagen, formulario, tabla, etc.).

a) Atributos opcionales

width = { *n* | *p* %}: Fija la anchura de la tabla en *n* píxeles o *p%* de la anchura disponible (por defecto: ajuste automático según contenido de las celdas). (No es un atributo desaconsejado, pero se aconseja que se utilice en su lugar una hoja de estilo).

cellspacing = { *n* | *p* %}: Fija el espacio entre las celdas (y entre las celdas y el borde exterior) en *n* píxeles o *p%* de la anchura disponible.

cellpadding = { *n* | *p* %}: Fija el espacio entre el borde de la celda y su contenido, en *n* píxeles o *p%* de la anchura disponible para la celda. (Los valores asignados a estos dos atributos pueden entrar en conflicto con el asignado a width; cuyos valores por defecto dependen del navegador.)

border = *n*: tabla debe tener un borde de (marco) de *n* píxeles (por defecto: sin marco).

frame = "{*[void]* | *above* | *below* | *hsides* | *lhs* | *rhs* | *vsides* | *box* | *border*} ": [4.0] Especifica qué lados del marco serán visibles: ninguno, superior, inferior, superior e inferior (horizontales), izquierdo, derecho, izquierdo y derecho (verticales), todos (*box* y *border*).

rules = "{*[none]* | *groups* | *rows* | *cols* | *all*} ": [4.0] Especifica qué líneas aparecerán entre las celdas de la tabla: ninguna, entre grupos de celdas y columnas (establecidas los primeros con **<thead>**, **<TFOOT>** y **<tbody>**, y las segundos con **<COLGROUP>**), entre filas, entre columnas, entre celdas.

summary = *"texto"*: [4.0] Establece un texto como resumen de la finalidad y estructura de la tabla, para navegación no visual (braille, voz, etc.). (Aconsejado por razones de accesibilidad.)

align = {[*left*] | *center* | *right*...}: Fija el ajuste horizontal de la tabla en la ventana o en el bloque en el que está insertada. Desaconsejado a favor de las hojas de estilo.

bgcolor = *"color"*: Fija un color para el fondo de la tabla. Desaconsejado a favor de las hojas de estilo.

Nota

Para mantener la compatibilidad con versiones anteriores de HTML, *border* = *"0"* implica *frame* = *"void"* y *rules* = *"none"*; otro valor de *border* implica *frame* = *"border"* y *rules* = *"all"*; *border* como atributo booleano (sin valor asignado, admisible en versiones anteriores) implica algún valor por defecto mayor que cero para *border, frame* = *"border"* y *rules* = *"all"*.

2) Elementos propios de <TABLE>

a) Título: el elemento **<caption>**

<CAPTION>Título**</CAPTION>**

Establece un título para la tabla. Opcional pero único en la tabla, debe seguir inmediatamente a la etiqueta inicial de **<TABLE>**.

- Atributo opcional:

align = *{[top]* | *bottom* | *left* | *right*}: Fija su situación con respecto a la tabla: encima, debajo, a la izquierda o a la derecha. Desaconsejado a favor de las hojas de estilo. (Algunos navegadores nunca han reconocido los valores *left* y *right*.)

b) Filas: el elemento **<TR>**

<TR>elementos TH y/o TD**</TR>** (*table row*).

- Crea una nueva fila de celdas.

- Puede llevar atributos de alineación del contenido de las celdas.

c) Celdas: los elementos **<TH>** y **<TD>**

<TH>Encabezado**</TH>** (*table header*)

<TD>Dato**</TD>** (*table data*)

Fijan el contenido de una celda como cabecera de fila o columna (se suele mostrar en negrita y centrado), o como dato (dato numérico, texto, imagen, formulario, tabla, etc.), de manera que el navegador pueda presentarlas de modo diferente sin recurrir a hojas de estilos.

Además de atributos de alineación del contenido de las celdas, puede llevar otros atributos como *headers, scope, abbr axis, colspan, rowspan, width, height*, etc.

2.7.8. El elemento <HR>

<HR></HR>

Pone una línea o barra (*horizontal rule*) entre secciones de texto.

Puede llevar atributos de núcleo y de acontecimientos básicos, más los siguientes (todos desaconsejados en favor de las hojas de estilos).

**1) El elemento
**

**
</BR>**

- Fuerza un corte de línea (*line break*) en la posición de este elemento.
- Puede llevar atributos de núcleo.

2.8. El estándar de escritura XHTML 1. 0

XHTML responde al acrónimo eXtensible HTML, es decir, a una revisión o mejora de las especificaciones del HTML.

Esta nueva orientación del lenguaje HTML ha surgido de la necesidad de *permitir una correcta interpretación de la información independientemente del dispositivo desde el que se accede a ella*.

El paso del HTML a XHTML supone la aceptación de unos requisitos básicos en cuanto a la estructuración y definición del documento web que aseguren que este documento, al respetar los estándares, resulta legible para cualquier dispositivo presente o futuro.

Este estándar tiene un punto clave: la separación del **contenido** y la **apariencia**.

El **contenido** se realiza en XHTML, que mantiene básicamente las mismas reglas que el HTML.

Así, en el lenguaje XHTML escribiremos el texto de nuestra página e incluiremos las imágenes y los enlaces a otras páginas, etc. Pero no especificaremos el tipo de letra que se debe utilizar ni el tamaño de las imágenes ni los colores ni ningún elemento de presentación.

La **apariencia** concreta de la página, los tipos de letra, los colores, el tamaño y la colocación de las imágenes se concretan en las denominadas hojas de estilo (**CSS**). Un mismo documento XHTML puede tener preparados diferentes CSS según, por ejemplo, el aparato que debe visualizar el contenido (diferentes ordenadores, móviles, impresoras, etc.).

2.8.1. Estructura de un documento XHTML

Un documento XHTML, para ser válido, debe tener una estructura mínima:

1) Declaración del tipo de documento

En la primera línea de un documento XHTML debe aparecer siempre la declaración del tipo de documento, dentro de la etiqueta **`<!DOCTYPE>`**: Es un requisito **obligatorio**. Sin esta declaración un documento XHTML **no es válido**, ya que el navegador o dispositivo desconocerá cuáles son las reglas que lo deben ayudar a interpretarlo correctamente.

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

2) Elemento raíz e idioma

Un documento XHTML ha de tener siempre un elemento raíz. Este elemento raíz se abre con la etiqueta **`<html>`** y se cierra con **`</html>`** poniéndola siempre al final del todo del documento. Inmediatamente después de la declaración del tipo de documento se abre el elemento raíz con la etiqueta **`<html>`**, que debe contener los atributos que hacen referencia al espacio de nombres y al idioma.

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

El atributo que hace referencia al espacio de nombres del XHTML (*xml namespace*) informa al navegador de lo que significa cada etiqueta y atributo del lenguaje XHTML. El segundo atributo corresponde al idioma. No especificar el idioma del contenido del documento va en detrimento de la accesiblidad y de una correcta interpretación de los contenidos.

3) Cabecera del documento

La cabecera de un documento XHTML se abre con la etiqueta **`<head>`** y se cierra con **`</head>`**. Entre estas dos etiquetas se sitúa toda una serie de información relativa al documento: tipo de contenido, título, metadatos, enlaces, referencias, etc.

El único elemento, sin embargo, que debe aparecer obligatoriamente en la cabecera es el título del documento, que se abre con la etiqueta **`<title>`** y se cierra con **`</title>`**. También son requeridos, aunque no es imprescindible, la declaración del tipo de contenido y el juego de caracteres.

4) Tipo de contenido

El tipo de archivo y el conjunto de caracteres del documento XHTML es requerido, pero no obligatorio, en la cabecera del documento. Si no se declara el tipo de contenido, el documento XHTML no será *completamente* válido. El informe de validación advertirá de la ausencia de declaración de la codificación de caracteres del documento.

El valor -texto/html- corresponde al tipo MIME del archivo y es necesario para que el navegador interprete y procese correctamente el archivo como documento web.

El segundo valor está relacionado con la internacionalización del contenido del documento, es decir, la representación correcta del alfabeto, los símbolos y la puntuación específica de la totalidad de los idiomas del mundo. El W3C recomienda la codificación **UTF-8** como juego de caracteres para documentos web por diferentes razones, como puede ser el hecho de que soporta un gran número de caracteres de diferentes idiomas, y por ser compatible entre diferentes contenidos.

5) Título del documento

El título del documento se abre con la etiqueta **`<title>`** y se cierra con la etiqueta **`</title>`**.

El título del documento es el que aparecerá en la parte superior de la ventana del navegador. También aparece en la pestaña del navegador en la que se encuentra abierto el documento (en el caso de los navegadores que soportan pestañas), y es el valor que identifica la web en el listado cuando se guarda como marcador (*bookmark*).

```
<head>
meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Título del documento</title>
</head>
```

6) Cuerpo del documento

El cuerpo incluye el contenido visible del documento XHTML en el navegador. Se inicia con la etiqueta **<body>** y se cierra con la etiqueta **</body>**. Cualquier elemento, carácter, etc. del contenido que no se encuentre entre las dos etiquetas **<body>**...**</body>** invalidará el documento.

```
<body>
<!-- inicio del contenido del documento -->
<!-- final del contenido del documento -->
</body>
```

2.8.2. Reglas para la validación correcta de la sintaxis del lenguaje XHTML

Existe una serie de reglas que un documento XHTML debe cumplir siempre para ser un documento válido. Es muy importante tenerlas presentes. Son las siguientes:

1) Un documento XHTML **requiere una declaración del tipo de documento** para ser correctamente interpretado por los navegadores y, por lo tanto, susceptible de ser validado.

Esta declaración se conoce como DTD y aparece en el prólogo del documento dentro de la cláusula del tipo de documento (**<!DOCTYPE>**), y sería como un tipo especial de regla gramatical y de sintaxis que cualquier documento basado en XHTML debe respetar.

La declaración del DTD debe ser siempre la primera línea del documento XHTML y su contenido ha de ser el siguiente:

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

Sin esta declaración un documento XHTML no es válido, ya que el navegador o dispositivo desconocerá cuáles son las reglas que lo deben ayudar a interpretarlo correctamente.

2) Un documento XHTML **debe tener todas las etiquetas, atributos y valores en minúsculas**.

Por ejemplo, se deberá poner **`<body>`** en lugar de **`<BODY>`**, o **`<h2>`** en lugar de **`<H2>`** y por lo tanto, por ejemplo, **`<table>`**, **`<Table>`** y **`<TABLE>`** serán interpretados como elementos diferentes.

3) Un documento XHTML, a diferencia del HTML, **requiere la presencia de la etiqueta `<html>`** y, además, ésta debe incluir los atributos *xmlns*, *xml:lang* y *lang*:

a) El atributo *xmlns* contiene el valor: "http://www.w3.org/1999/xhtml".

b) Los atributos *xml:lang* y *lang* corresponden al idioma específico del documento, que en este caso será *ca* (catalán).

La etiqueta **`<html>`** del documento quedaría, pues, de la siguiente manera:

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

4) Todos los elementos de marcado de un documento XHTML se deben cerrar con la etiqueta correspondiente y todos los atributos han de quedar entre comilla. Fijaos en los siguientes ejemplos:

```
<p>texto, <em>texto enfatizado</em></p>
```

De la misma manera que todos los elementos de marcado del documento XHTML se deben cerrar con la etiqueta correspondiente, **los valores de un atributo han de ir siempre entre comillas.** Observad el siguiente ejemplo:

```
<p id="texto">texto con un atributo</p>
```

5) Los elementos vacíos como el salto de línea `<br>` deben cumplir también esta norma. Así, la etiqueta **`<br>`**, válida en HTML, será válida en XHTML, añadiéndole una barra inclinada antes del signo de cierre: **`<br />`**. Otros elementos vacíos que resultan afectados por esta norma son: **`<col />`**, **`<img />`**, **`<input />`**, **`<link />`** y **`<meta />`**.

Ejemplo:

```
<img src="exemple.png" alt="exemple" title="exemple"
width="200" height="156" />
```

6) Los elementos del documento XHTML deben estar anidados correctamente, es decir, las etiquetas que abren y cierran los diferentes elementos han de estar en el orden correcto. Observad un ejemplo:

Ejemplo:

** **texto ****texto enfatizado****.** **

Se ha empezado con la etiqueta ** **, seguida de un elemento en línea **** que, más adelante, se cerrará con **** y, para acabar, se cerrará el inicio del párrafo con la etiqueta ** **

Por otra parte, la declaración del tipo de documento XHTML especifica un conjunto de reglas de anidación de los elementos. Según estas reglas no todos los elementos pueden ser anidados por otros elementos:

a) Un elemento *a* no puede contener otros elementos *a*.

b) El elemento *pre* no puede contener los siguientes elementos: *img, object, big, small, sub, sup, font, applet o basefont*.

c) Un elemento *button* no puede contener los siguientes elementos: *input, select, textarea, label, button, form, fieldset, iframe o isindex*.

d) Un elemento *label* no puede contener otros elementos *label*.

e) Un elemento *form* no puede contener otros elementos *form*.

2.9. Validación HTML 4.0 y XHTML 1. 0

La validación es un concepto importante en el desarrollo de páginas webs. La validación es el proceso de comprobar el código de un documento HTML para garantizar que su codificación es la correcta (sintaxis correcta y no existencia de etiquetas o atributos no estándares) antes de la visualización con un navegador.

2.9.1. Web de validación del W3C

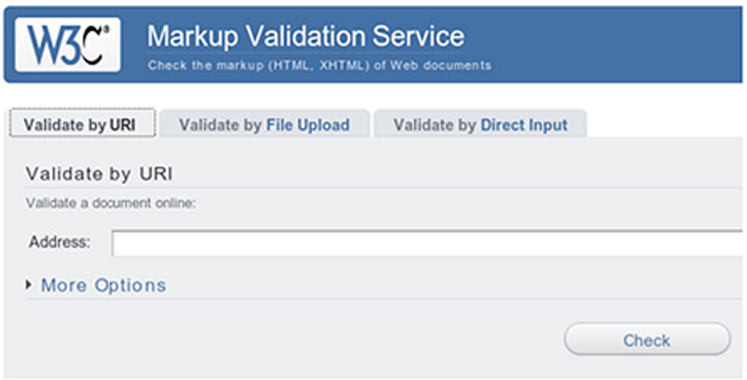
El W3C ofrece un servicio gratuito de validación de documentos HTML, XHTML y CSS. Este servicio, conocido como Markup Validation Service incluye tres variantes:

1) Por medio de una URI válida: http://validator.w3.org/#validate_by_uri.

2) Subiendo un archivo: http://validator.w3.org/#validate_by_upload.

3) Entrada directa: http://validator.w3.org/#validate_by_input.

Herramienta web de validación del W3C

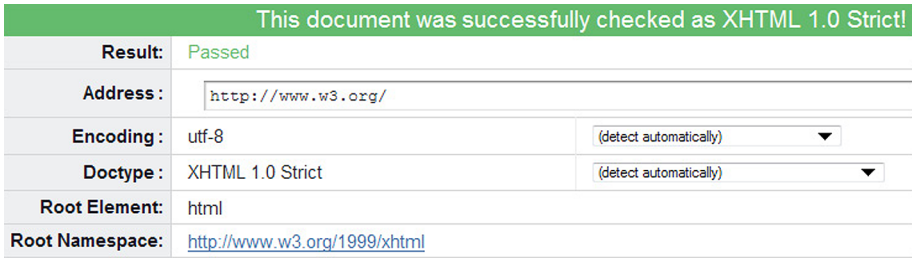


Validar correctamente los documentos HTML y XHTML es importante, pues una validación correcta permite a los navegadores interpretar adecuadamente el documento. Un documento no válido supone errores de sintaxis en el documento que repercuten tanto en su significado (la parte semántica del documento), como en su presentación (un documento no válido presentará problemas en la hoja de estilo asociado).

El proceso de validación pasa por utilizar un DOCTYPE en cada cabecera del documento. El **DOCTYPE** es el conjunto de reglas que determina la estructura de los documentos y, por lo tanto, le indica al validador cómo debe interpretar correctamente el documento.

El validador *lee* el código del documento y localiza los posibles errores, siempre teniendo como referencia las especificaciones del **DOCTYPE** asociado al documento.

Resultado de la validación



Si el documento ha pasado con éxito la validación, el W3C ofrece un fragmento de código XHTML que muestra el icono de validación del documento.

Código XHTLM que muestra la validación

"valid" Icon(s) on your Web page

To show your readers that you have taken the care to create an interoperable Web page, Web page:

```
<p>
  <a href="http://validator.w3.org/check?uri=referer"><img
      src="http://www.w3.org/Icons/valid-xhtml10"
      alt="Valid XHTML 1.0 Strict" height="31" width="88" /></a>
</p>
```

```
<p>
  <a href="http://validator.w3.org/check?uri=referer"><img
      src="http://www.w3.org/Icons/valid-xhtml10-blue"
      alt="Valid XHTML 1.0 Strict" height="31" width="88" /></a>
</p>
```

2.9.2. Tidy

Tidy: http://www.w3.org/People/Raggett/tidy/. Es una aplicación que permite **limpiar y corregir** el código para obtener un resultado **válido**. Algunos de los errores que esta aplicación permite corregir son:

- Etiquetas mal cerradas o ausencia de cierre de etiquetas.
- Indentación del código para una mejor legibilidad.
- Codificación de un juego de caracteres a otro.

Existe una versión web de la aplicación, Tidy online (http://infohound.net/tidy/), que facilita el uso de Tidy sin necesidad de instalar la herramienta en el sistema operativo.

El W3C dispone también de un servicio Tidy, más simple que el anterior: http://cgi.w3.org/cgi-bin/tidy.

2.10. HTML 5 y XHTML5

HTML 5 (*hypertext markup language*, versión 5) es la quinta revisión del lenguaje básico de la World Wide Web, HTML. HTML 5 especifica dos variantes de sintaxis para HTML: un "clásico" **HTML (texto/html)**, la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5, que deberá ser servida como **XML (XHTML) (*application*/xhtml + xml)**. Hay que señalar que es la primera vez que HTML y XHTML se han desarrollado en paralelo, y su planteamiento es el de un lenguaje mucho más semántico y más propicio para el desarrollo de aplicaciones web.

2.10.1. Nuevos elementos

El DOCTYPE en HTML 5 es el siguiente:

```
<!DOCTYPE html>
```

HTML 5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas ** ** y ****, pero tienen un significado semántico, como **<nav>** (bloque de navegación del sitio web) y **<footer>**.

Otros elementos proporcionan nuevas funcionalidades mediante una interfaz estandarizada, como los elementos **<audio>** y **<video>**.

<header>: Representa la sección de la cabecera.

<nav>: Elemento reservado para la navegación principal.

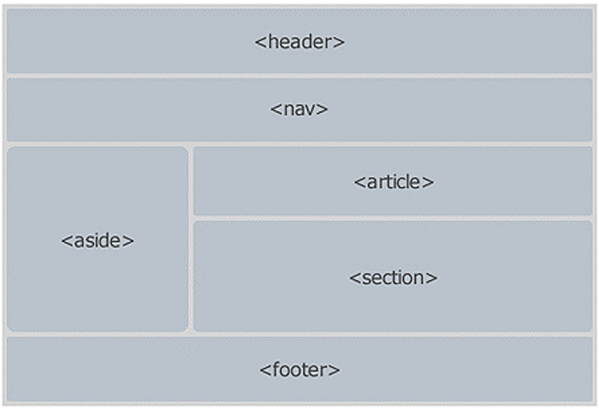
<section>: Elemento para una sección genérica de contenidos, posiblemente situado entre la cabecera y el pie de página (*footer*). Tiene el comportamiento de un div si no fuera por la separación de una parte del documento. Dentro de él pueden anidar otras etiquetas *section*, aparte de cualquier cantidad de etiquetas típicas.

<article>: Etiqueta que nos permite definir un fragmento de contenido como artículo. Ideal para bloques o diarios.

<aside>: Elemento que representa el contenido relacionado con el área principal del documento, expresado normalmente por las columnas laterales.

<footer>: Sección reservada al pie de página.

Ejemplo de estructura html5



2.10.2. Mejoras en el elemento

Algunos elementos de HTML 4.01 han quedado obsoletos, incluyendo elementos puramente de presentación, como **<font>** y ** **, ya que los efectos son manejados por el **CSS**. También hay un renovado énfasis en la importancia del scripting **DOM** para el comportamiento de la página web.

Páginas webs recomendadas

J. M. Nogales. "HTML.4.01". Universidad Carlos III de Madrid. Creación: 1998-08-16. Última actualización: 2007-02-22 <http://www.bib.uc3m.es/~nogales/cursos/html401.html> [en línea] [Última visita 23/04/2010]

Manual de HTML en catalán. <http://www.imaginaserveis.com/manuals/html/manual.html> [en línea] [Última visita 23/04/2010]

World Wide Web Consortium (W3C). <http://www.w3.org/> [en línea] [Ultima visita 23/04/2010]

Lecturas recomendadas

C. Egea (2007). *Diseño web para tod@s*. Materiales elaborados por Carlos Egea García, con la colaboración de Juan Carlos Ramiro Iglesias, Alicia Sarabia Sánchez. Barcelona: Icaria, ISBN 9788474266306 (vol. 1), 9788474269574 (vol. 2)

M. Pardo (2008). *Guía visual de creación y diseño web*. Madrid: Anaya Multimedia, ISBN 9788441523418

S. Shaefer (2008). *HTML, XHTML, and CSS Bible*. Indianapolis, Ind. : Wiley Pub 4th ed. ISBN: 9780470128619

3. Hojas de estilo en cascada (CSS)

3.1. Objetivos

En el presente apartado el objetivo que debe alcanzar cada alumno es obtener una idea general de qué son las hojas de estilo en cascada (HEC) y cómo poder implementarlas en una página web.

Asimismo, conocerá las ventajas que supone este tipo de programación o marcado web y las fuentes de las que poder obtener información para ampliar los conocimientos sobre este tipo de código.

3.2. Introducción

En los inicios de la creación web, el estándar **HTML** era más que suficiente para asumir contenido (el texto que uno quería transmitir a través de su sitio web) y diseño (el aspecto externo y la disposición de esos textos).

Con este sistema HTML, si queríamos dar colorido y estructura a nuestros textos, afrontábamos el diseño visual y las ubicaciones de los distintos elementos a partir de tablas o marcos, que aportaban la maquetación y la etiqueta **`<font>`** que, colocada junto a un texto

```
<font color = "black">En su artículo, <B>Lluís Codina</B> se centra en dos casos
de la <I><B>Web 2.0</B></I></font>
```

le proporcionaba distintos resaltes, colorido, tipo y características de las fuentes.

No obstante, hay que entender que, de hecho, el HTML no tenía vocación de ser un lenguaje "visual", sino más bien un lenguaje "conceptual". Y esto es porque su orientación era decirnos **qué era cada cosa dentro de un documento** y no **el aspecto que ésta tenía en el navegador**.

El HTML estaba ideado para indicar en cada tramo de texto **su valor estructural** y no **su aspecto externo** (pese a que se implementaron etiquetas que contravenían esta orientación principal, como **`<font>`** o **`<b>`**, empleadas principalmente en la visualización).

En la versión 3.0 de HTML ya aparecía la etiqueta **`<style>`**, que sería, por decirlo así, un antecedente de las hojas de estilo en cascada (HEC[8]). **`<style>`** en un principio se parecía mucho a lo que luego serían nuestras **CSS** o *cascade style sheets*.

(8)HEC = CSS.

A estas alturas ya se hacía evidente la necesidad de una filosofía como la que impera hoy en día: que una cosa es el valor estructural de un contenido (**HTML**) y otra el aspecto visual (**CSS**).

De este modo, y manteniendo esta separación entre forma y contenido, un mismo contenido se puede representar de múltiples maneras, si fuera necesario, tal como lo es a día de hoy a causa de la multitud de terminales (móviles, PC, consolas, televisores, impresoras, etc.).

Web recomendada

Podéis ver el atributo **media** en el W3C a través de los que se puede leer la información situada en la web o en formato web ubicado en cualquier soporte digital:

http://www.w3.org/TR/1999/REC-html401-19991224/types.html#type-media-descriptors.

Aunque nada impide que el contenido y el diseño sean realizados por una única persona, también se pone de manifiesto la separación que existe entre el programador (quien escribe el código y ubica el texto) y el diseñador (quien define el aspecto visual) en la estructura organizativa: de hecho, en las páginas webs suelen intervenir tanto programadores, creadores y selectores de contenido como diseñadores de grafismo electrónico.

La funcionalidad que otorga el que el diseño pueda ser trabajado por unos y el contenido y el código por otros permite una división del trabajo de cara al mantenimiento de la página web, y así la inversión de tiempo y dinero se reduce y la productividad mejora.

Por otro lado, ya que vimos que en un principio se mezclaban etiquetas de una y otra vertiente, el uso de la antigua etiqueta **`<font>`**, solución adoptada en un principio para dotar de características a las fuentes de texto, implicaba repetir muchas veces la misma información, lo que hacía que los ficheros fueran mucho más pesados en términos de KB, dado que cada uno de los caracteres de un archivo ocupa entre cuatro y dieciséis bits, según la tabla de caracteres empleada.

Puesto que el HTML estaba hecho para viajar por la Red, siempre era conveniente que su tamaño en KB fuera lo más reducido posible con el fin de que pudiera usarse con la rapidez oportuna y satisfactoria para el usuario.

CSS se introdujo por primera vez en torno a los años noventa, lo que no significa que los navegadores pudieran utilizarlo ya, pues necesitaban implementar en sus rutinas de programación el nuevo estándar y eso llevó cierto tiempo.

Hasta el año 2006, con la asunción por parte de Internet Explorer 7 de las HEC/CSS, ha habido todo un proceso de adaptación de estas hojas de estilo en cascada o (HEC/CSS) y de incertidumbre a la hora de adoptar unas características u otras propuestas por CSS, por parte de los diferentes navegadores.

Dado el carácter práctico de este material, no entraremos a comentar la parte histórica del desarrollo de HEC/CSS salvo cuando sea estrictamente necesario. Partiremos del momento actual y trataremos de mostrar cómo se pueden utilizar las hojas de estilo en cascada (HEC/CSS), de inmediato, lo más rápidamente posible y sin tener que pasar por la comprensión de todo el sistema de funcionamiento. Aunque, evidentemente, no se lograrán buenos resultados hasta que el estudiante profundice lo suficiente en el tema.

Lo haremos todo con los mismos medios con los que hemos programado hasta ahora el HTML,

1) un ordenador

2) que disponga de un navegador web (cliente) y

3) un editor de texto plano (aunque sería mejor usar algún editor de HTML[9] para evitar tener que escribir todas las etiquetas a mano).

(9)Tal como NVU o Dreamweaver.

Para realizar la fase de aprendizaje no es necesaria una conexión a Internet, pero se hará evidente más adelante que **se debe contar con una a la hora de poder probar los archivos en un servidor de Internet** (o al menos de red).

Presuponemos, por otro lado, que quien lee estos materiales tiene ya, aunque básicos, ciertos conocimientos del HTML y del lenguaje de marcado propuestos en los módulos anteriores, y que hasta cierto punto ya está acostumbrado a programar HTML de manera elemental con un editor de texto y un navegador (IExplorer, Firefox o incluso Safari u Opera).

3.3. Etiquetas básicas para el uso de hojas de estilo en cascada

3.3.1. Introducción

Comenzaremos viendo cómo se pueden insertar los estilos en un documento HTML de la manera más sencilla y luego iremos viendo cómo existen formas que, aunque son más complejas, en realidad no son más difíciles de uti-

lizar. Primero examinaremos la etiqueta **<style>** (3.3.2) y luego extraeremos las instrucciones de esta etiqueta a un archivo externo y enlazado con aquel HTML, con el que deberá actuar (3.3.3) mediante de la etiqueta **<link>** (3.3.4).

3.3.2. La etiqueta <style>

La etiqueta **<style>** nos puede ayudar a entender de manera sencilla cómo funcionan las hojas de estilo en cascada (HEC/CSS), sin la necesidad de "salir" del archivo HTML de trabajo, esto es, trabajando "dentro" del propio archivo de trabajo HTML.

Luego, más adelante, tal como hemos comentado, veremos cómo se puede delegar el contenido HEC/CSS de este archivo en una hoja secundaria HEC/CSS y separada del archivo HTML de trabajo que estamos manejando, pero, de momento, aprenderemos la sintaxis de HEC/CSS a partir de **<style>**, que resultará más cómodo para la comprensión del estudiante.

Partamos de un archivo sencillo en HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<head>
<title>Hojas de Estilo en Cascada.</title>
<meta http-equiv="content-type" content="text/HTML; charset=iso-8859-1">
</head>
<body>
<p>Hola Mundo</p>
</body>
</HTML>
```

Haremos clic aquí para observar el resultado de este archivo en el navegador.

Pero así queda muy soso, debemos reconocerlo. Nos gustaría, por ejemplo, que estuviera en rojo y resaltado, para empezar a sacar partido a nuestros conocimientos de HEC/CSS. Para que el navegador pueda interpretar nuestras intenciones correctamente, necesitamos añadir al archivo las siguientes instrucciones o líneas de código:

Primero debemos marcar el elemento al que queramos añadir estilo. Esto lo hacemos añadiendo a su etiqueta HTML un atributo (cuyo nombre servirá para identificarlo) denominado **class** y cuyo valor será aplicado. Así:

```
<p class = "miEstilo1">Hola Mundo</p>
```

Luego debemos indicar las instrucciones que definirán el estilo del texto "Hola Mundo" a partir de la palabra clave (miEstilo1) que le hemos asignado.

Estas instrucciones HEC/CSS que indicarán el estilo que se aplicará a nuestro texto las colocaremos entre las etiquetas **<style></style>** y preferiblemente en el apartado HEAD.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<head>
<title>Hojas de Estilo en Cascada.</title>
<meta http-equiv="content-type" content="text/HTML; charset=iso-8859-1">

<style> type="text/CSS">

p.miEstilo1{

color:red;
font-weight:bold;

}
</style>

</head>
<body>
<p class = "miEstilo1">Hola Mundo</p>
</body>
</HTML>
```

Para referenciar en el código que indica el estilo el párrafo que etiquetamos como <p class = "miEstilo1"> usaremos bien la notación .miEstilo1 o bien p.miEstilo1 indistintamente, son equivalentes, aunque una (.) tiene un sentido más general que la otra (p.). Para agrupar los distintos atributos y valores de un mismo estilo usaremos las llaves {}.

Cada una de las definiciones de estilo usará un atributo –cualidad abstracta aplicable al texto– y un valor – la concreción de esa cualidad en un dato concreto– escritos de la siguiente forma **atributo**: **valor**.

En conjunto, una instrucción sencilla como poner el texto de color rojo y en negrita quedaría de la siguiente manera:

```
p.miEstilo1{

color:red;
font-weight:bold;

}
```

Haremos clic aquí para observar el resultado de este archivo en el navegador.

Es preferible, para evitar ambigüedades, la notación p.miEstilo1 a sencillamente .miEstilo1.

La segunda nos permitiría compartir las características del estilo con otro tipo de etiquetas, siempre que incluyesen el mismo atributo class = "miEstilo1", por ejemplo:

```
<span class = "miEstilo1">Hola Mundo</a>
```

Y adoptaría las mismas características que el párrafo **<p>** con el que tratamos cualquier etiqueta a la que se lo aplicáramos. Aunque este efecto de generalizar un estilo puede ser deseable en algún momento, muchas veces será mejor ser más concreto y preciso, pues, a la larga, resulta más cómodo el mantenimiento de la página si sabemos con exactitud y precisión a qué etiquetas va a afectar cualquier variación o cambio.

3.3.3. Estandarización de <style>

El World Wide Web Consortium (W3C) es el organismo encargado de estandarizar el uso del lenguaje de marcado en sus distintos sublenguajes. La mejor manera de saber cómo y qué puede hacer una etiqueta o elemento del lenguaje es acudir a él y ver cómo está definido allí; de este modo podemos saber cómo usarlo y qué atribuciones y alcance tiene, incluso si está en desuso. Es la fuente de información sobre las etiquetas HTML más fiable.

En el W3C [http://www.w3.org/TR/1999/REC-html401-19991224/present/styles.html#edef-STYLE] el elemento **<style>** se define de la siguiente manera:

```
<!ELEMENT <STYLE> - %<style>Sheet  - <style> info - >
<!ATTLIST <STYLE> %i18n; - lang, dir, for use with title -
type %ContentType;  #REQUIRED - content type of <style> language -
media %MediaDesc; #IMPLIED - designed for use with these media -
title %Text; #IMPLIED - advisory title - >
```

Que, en resumen, significa que sólo necesita indicar el tipo del texto que contiene (type %ContentType; #REQUIRED), que puede ser CSS o HTML. En este caso contiene CSS pues lo usamos para otorgar estilo a las distintas etiquetas HTML.

La etiqueta **<style> </style>** debe colocarse en el encabezado del archivo HTML, en la sección HEAD, entre las etiquetas **<head>** y **</head>**. Aunque funciona ubicada en el cuerpo, BODY, de un archivo HTML, no debe colocarse en ese lugar, debe ir siempre en el encabezado o HEAD.

Ejemplo

Véase como ejemplo el siguiente problema. http://www.desarrolloweb.com/articulos/1167.php

A la hora de cargar un archivo el navegador comienza por **<head>**, lo que supone una ventaja en algunas ocasiones a la hora de programar (véase ejemplo).

3.3.4. La etiqueta <link>

<style> es una etiqueta que podemos usar para un proyecto pequeño. Un problema que presenta **<style>** es que deberá incorporarse este etiquetado en todas y cada una de las páginas que hayan de implementar los estilos que contiene.

De este modo, si el código debe repetirse en cada página y las páginas son muchas, se reproduce el problema que teníamos anteriormente con el uso de **<font>**, pues la repetición de código aumenta los archivos de texto y de nuevo, en un proyecto grande, puede tener mucho coste en KB repetir una y otra vez las mismas instrucciones.

Por lo tanto, el modo como las distintas páginas pueden compartir este código de estilo es extraer las definiciones de estilo a una página externa, en lugar de repetir en cada una de las nuestras el código **<style>** y después referenciar la hoja externa CSS mediante un enlace en todas y cada una de las páginas de nuestro trabajo.

El archivo externo que usaremos llevará la extensión .CSS y usaremos **<link>** para referenciar dicho archivo. En el gráfico adjunto podemos observar las dos maneras de implementar estilos, con **<style>** y con **<link>**:

La etiqueta **<link>** nos permitirá hacer la referencia al archivo externo desde todas y cada una de las páginas. Las ventajas de este procedimiento son evidentes:

1) hay **menos código**, al evitarse las repeticiones;

2) además, **entre todos los archivos de una misma web se puede compartir estilos semejantes**, hecho muy ventajoso si tratamos de dar un estilo unificado a todo un espacio web;

3) a la hora **de renovar o mantener los estilos sólo deberemos trabajar con la hoja de estilo**, sin la necesidad de retocar el HTML que incluyen el resto de las páginas;

4) junto a esto, y como podremos ver más adelante en este documento, **podremos seleccionar el tipo de medio al que está orientada la página** (navegador, impresora, móvil, videoconsola, etc.) y crear distintas hojas de estilo para la misma documentación, de modo que estas hojas creadas permitan que el contenido se visualice de distintas maneras acorde al medio al que ha accedido la página HTML.

El código de nuestra nueva modalidad para aplicar estilo quedaría así:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<head>
<title>Hojas de Estilo en Cascada.</title>
<meta http-equiv="content-type" content="text/HTML; charset=iso-8859-1">
<link href="My.CSS" rel="<style>sheet" type="text/CSS" />
</head>
<body>
<p class = "miEstilo1">Hola Mundo</p>
</body>
</HTML>
```

Para que **`<link>`** se ejecute con precisión, deberemos rellenar tres atributos:

1) href, cuyo contenido es una URL, la URL del archivo que servirá el estilo CSS. En nuestro caso se encuentra en el mismo directorio que las hojas con HTML y recibe el nombre de My.CSS. Aunque es habitual incluirlo en un directorio específico (/CSS), de modo que la estructura del sitio web esté más ordenada y sea más fácil de mantener.

2) rel, el tipo de relación que hay entre la hoja referenciada (My.CSS) y la actual, donde se incluye la referencia.

3) type, el tipo de archivo (texto) y la especificación de qué uso tiene el texto (CSS).

El archivo My.CSS contendrá únicamente lo que antes ubicamos entre las etiquetas **`<style>`** y **`</style>`**.

Código del archivo My.CSS:

```
/* CSS Document */
p.miEstilo1{
color:red;
font-weight:bold;
 }
```

Haremos clic aquí para observar el resultado de este archivo en el navegador.

El elemento **<link>** no tiene únicamente el uso que le hemos dado, tiene otros que podemos ver definidos en el W3C.

3.3.5. Estandarización de <link>

La definición de LINK es la siguiente:

```
<!ELEMENT LINK - O EMPTY -- a media-independent link -->
<!ATTLIST LINK
%attrs; -- %coreattrs, %i18n, %events --
charset %Charset; #IMPLIED -- char encoding of linked resource --
href  %URI; #IMPLIED -- URI for linked resource --
hreflang  %LanguageCode; #IMPLIED -- language code --
type  %ContentType; #IMPLIED -- advisory content type --
rel  %LinkTypes;  #IMPLIED -- forward link types --
rev%LinkTypes;  #IMPLIED -- reverse link types --
media  %MediaDesc;#IMPLIED -- for rendering on these media -- >
```

En la lista de atributos encontramos la posibilidad de asignar un juego de caracteres o **charset**; la opción para referenciar un archivo mediante una URI, **href** ; el lenguaje del enlace, **hreflang**; tipo de contenido con **type** ; las relaciones de ida y vuelta del enlace (**rel**, **rev**) y, por último, el tipo de medio, **media** (pantalla, impresora, etc.), que leerá y representará el contenido enlazado.

Todos estos atributos son utilizados para aportar información a los motores de búsqueda y a los navegadores cliente, aunque aquí no iremos más allá del uso que nos interesa, que son las hojas de estilo en cascada que hemos visto anteriormente.

3.4. Atributos básicos para el uso de hojas de estilo en cascada

3.4.1. Los atributos id y class

Id y **class** son atributos semejantes, pero ligeramente diferentes; su uso general es el que hemos comentado: sirven como identificadores para la aplicación del estilo y posicionamiento que aportan las hojas de estilo en cascada (HEC/CSS).

Funciona igual:

```
p.miEstilo2{                    que                    #miEstilo3{

color:red;                                             color:red;
```

```
}                                                    }

<p class="miEstilo2">                                <p id="miEstilo3">
```

Tal como se puede comprobar en el ejemplo.

Observad, no obstante, que la notación de CSS es ligeramente diferente si se trata de un identificador (#id) o de una clase (p.class).

Usaremos los identificadores (id = "miEstilo3") con elementos únicos en el documento cargado en el navegador, mientras que reservaremos la clase (class = "miEstilo2") para aquellos estilos que debamos utilizar repetidamente dentro del documento HTML.

Estos dos atributos van a ser esenciales a la hora de sacar partido a las CSS.

3.4.2. Funcionamiento de los atributos de estilo

En la notación CSS incluimos entre las llaves { y } los atributos de una etiqueta de HTML, asignándoles un valor:

```
.nombreEstilo{
Atributo1 : valor1 ;
Atributo2 : valor2 ;
}
```

Estos atributos y sus posibles valores vienen definidos en la especificación de CSS.

Cada etiqueta o *tag* tiene sus atributos. Quizá los atributos o los posibles valores de éstos que más nos interesen de manera inmediata sean los referidos al tipo y aspecto de las fuentes, así como los referidos a cualidades otorgables al texto.

No obstante, la potencia de CSS es grande cuando hacemos uso de todos estos atributos, pues, a la hora de efectuar un diseño impactante y/o dinámico, se puede operar sobre etiquetas como **`<div>`** y **`<span>`** (de las que hablaremos más adelante), que resultan muy genéricas y permiten desde crear un cuadro y ubicarlo hasta componer complicados efectos de texto o de interacción como menús desplegables o que cambian de color al pasar el puntero sobre ellos, efectos que hasta hace poco debían ser programados con un complicado código de **JavaScript**. Ahora podemos conseguirlos fácilmente con HEC/CSS.

Aunque los listados que proporcionamos del W3C son una información muy precisa y detallada sobre los diferentes atributos, en realidad sólo deberá ser consultada eventualmente, pues los editores WYSIWYG de HTML ya incorpo-

ran listados completos y documentados de atributo y valores que nos ayudarán a la hora de la composición del documento. No obstante, siempre es importante disponer de una fuente de información completa como el W3C.

3.5. Cómo tratar las etiquetas con CSS (diseño y posicionamiento básicos)

3.5.1. Introducción

Como hemos aprendido al ver HTML, las etiquetas responden a conceptos de un documento, párrafo, título e imagen; sin embargo, en CSS deberemos concebirlas como un área de dibujo, como un cuadrado de papel (etiqueta) que podemos pegar aquí o allá.

Esta conceptualización las unifica a todas y nos ayuda a concebir mentalmente nuestros documentos. Imaginemos que trabajamos con cuadros que podemos estirar, encoger, ampliar o reducir y acoplar o solapar, incluso jugar con su opacidad y transparencia. Cada uno contendrá un elemento distinto, un color sencillamente, texto, imagen, símbolos o dibujos, aunque no deberemos fijarnos en lo que contiene, sino en concebirlos como un cuadro o etiquetas.

Imaginemos todos los elementos que deseamos disponer en un página como si los tuviéramos sobre la mesa escritos en papel transparente de colores y recortados al tamaño que nos interesa en cada caso.

Con paciencia los iríamos colocando unos junto a otros, de manera que al fin compusieran una página unitaria semejante a la de una revista o un diario, incluso alguna tan sencilla como la de un libro, en nuestro caso una web.

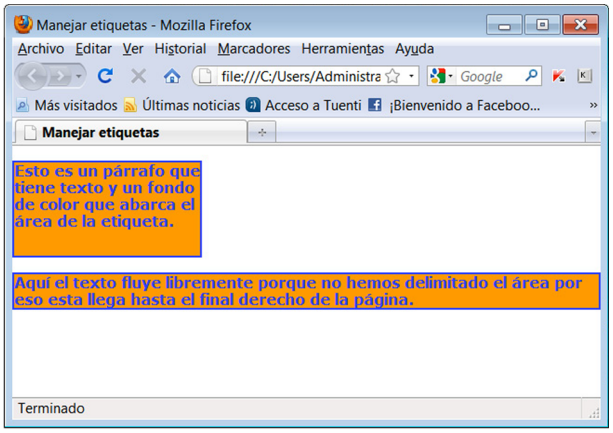
Pues eso es lo que vamos aprender a hacer, pero ahora con material electrónico.

3.5.2. "Elasticidad" de las etiquetas

Veamos un ejemplo de sencillos cuadros con texto en un archivo HTML.

```
p.miEtiqueta1{
        font-family:Verdana, Geneva, sans-serif;
        font-size:16px;
        font-weight:bold;
        color:#33F;
        background:#F90;
        border:#33F solid 2px;
        width:200px;
        height:100px;
}
```

```
p.miEtiqueta2{
        font-family:Verdana, Geneva, sans-serif;
        font-size:16px;
        font-weight:bold;
        color:#33F;
        background:#F90;
        border:#33F solid 2px;
}
```



Con CSS controlaremos dos aspectos fundamentales,

1) la ubicación de un área en la página web, y
2) el estilo visual que tendrá esta área.

También se incluye en estas posibilidades la ubicación y el estilo de los elementos que contienen, dentro del área en este caso los distintos textos que podemos observar.

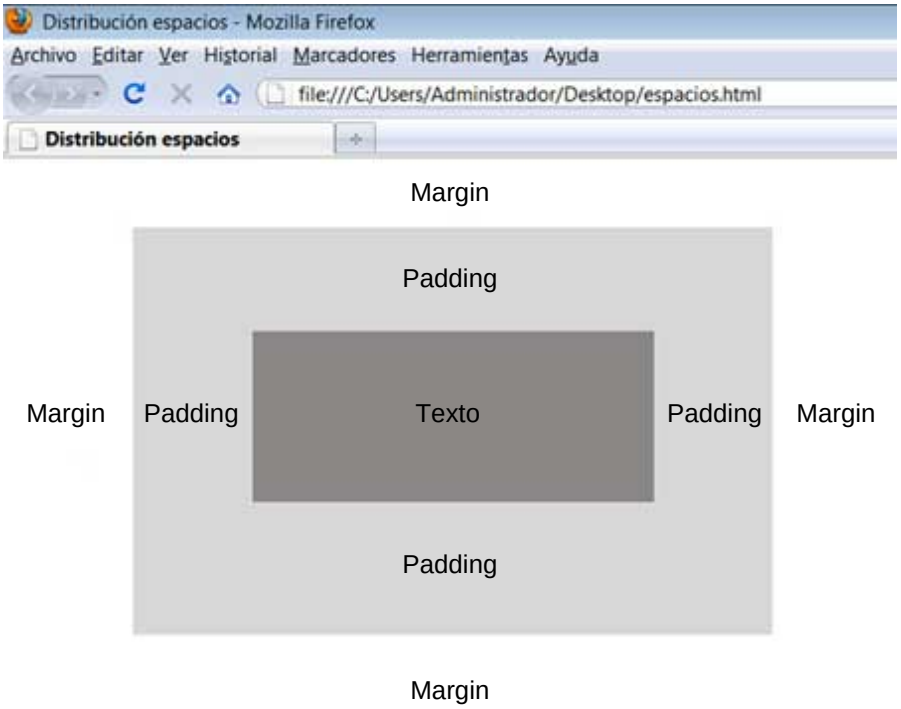
Interpretación de los navegadores

En todo caso siempre habrá que ir con cuidado porque no todos los navegadores interpretan exactamente igual todos los atributos y esto puede ocasionar que lo que se ve de una manera en un navegador sea diferente en otro.

En nuestro ejemplo vemos dos cuadros que tienen aspectos semejantes pero que no se comportan exactamente igual uno y otro, a uno se le ha marcado el área y al otro se le ha dejado fluir –que es el modo natural de comportarse de los elementos en HTML.

3.5.3. Los espacios invisibles

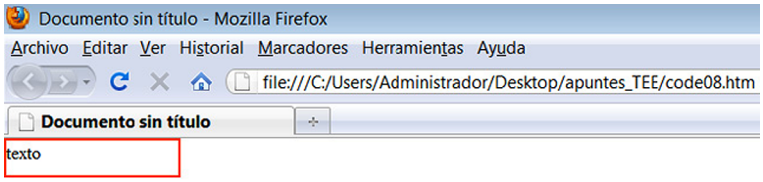
Veamos cómo funciona el posicionamiento de un elemento. En el siguiente gráfico podemos ver los atributos de posicionamiento más básicos que nos pueden ayudar. Aquí comentaremos las líneas básicas, y el estudiante interesado deberá, con el estudio y, sobre todo, la práctica, completar sus destrezas, pues el tema es lo suficientemente amplio y complejo como para poder tratarlo aquí en extensión.



Hemos situado un único elemento, que denominaremos <p class="parrafo_posicionado">, en el que aparece la palabra *texto* representada por el cuadro interior. Sus atributos se describen como sigue:

```
p.parrafo_posicionado{
margin : rodea el cuadro que contiene nuestro texto. Podemos establecerlo de manera global, con
"margin", o de manera más específica, con "margin-top", "margin-left", "margin-right" o
"margin-botton".
padding: distancia desde el filete del cuadro hasta los elementos que contiene (texto o imagen
habitualmente). Podemos establecerlo de manera global, con "padding", o de un modo más específico,
con "padding -top", "padding -left", "padding -right" y "padding-botton".
}
```

Veamos esto puesto en práctica en un archivo HTML.



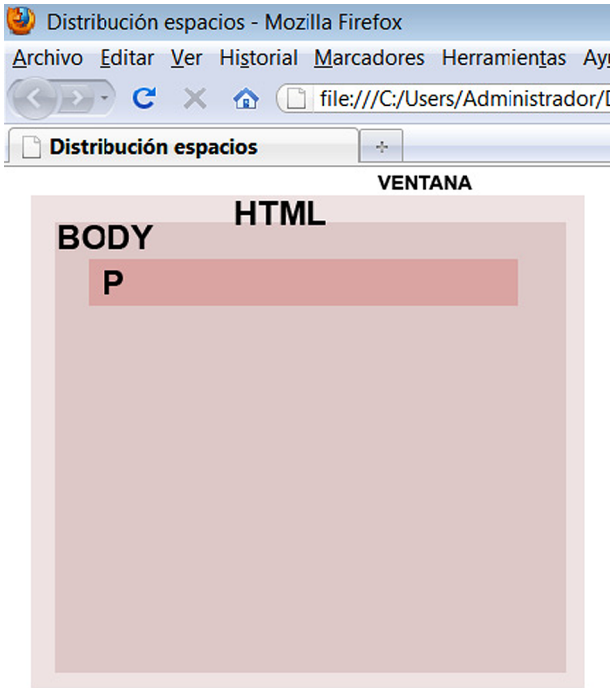
Observaremos que el cuadro rojo que representa el filete de nuestra etiqueta queda pegado a la parte superior y al margen izquierdo. Esto no es casual ni automático, sino que se ha conseguido modificando previamente las etiquetas HTML y BODY en la hoja CSS para que sus *margin* y *padding* sean de 0 píxeles,

```
html, body{
  margin: 0px;
  padding: 0px;
}
```

De no hacerlo de esta manera, nos encontraríamos con un margen que no esperábamos, ya que nuestro elemento **<p>** está incluido como mínimo entre las etiquetas **<html></html>** y **<body></body>**, anidado del siguiente modo:

```
<html>
<body>
<p class = "parrafo_posicionado"
</body>
</html>
```

Y debemos tener presente que la estructura del documento sería la siguiente:



Así, resulta necesario considerar que **<html>** y **<body>** también tienen sus *margin* y *padding* y que los navegadores no siempre tratan este aspecto de la misma manera, por si fuese poco.

Veamos ahora el siguiente archivo HTML.

Píxeles extras

Señalamos esto porque la sorpresa más inmediata del programador es ver que siempre aparece una serie de píxeles extra que no se sabe muy bien de dónde ha salido y que proporciona algunos dolores de cabeza.


```
p.parrafo_posicionado{
        height: 30px;
        width: 150px;
        border: red solid 2px;
        margin: 0px;
        padding: 0px;
}
```

```
p.parrafo_posicionado2{
        height: 30px;
        width: 150px;
        border: red solid 2px;
        margin-top: 31px;
        margin-left: 155px;
        padding-top:10px;
        padding-left:10px;
}
```

Con el archivo en pantalla, y a ser posible también el código **CSS**, observamos lo siguiente:

1) el primer párrafo (texto1) está pegado al margen, y
2) el segundo párrafo (texto2) se ha separado del margen izquierdo según el atributo que le asignamos en la hoja CSS en margin-left 155px. Esto es intuitivo y fácil de ver, pero el atributo **margin-top** no toma como referencia, como sería esperado, el borde superior de la ventana, sino el primer cuadro de texto (**texto1**) para desplazarse 31px. Así que su desplazamiento hacia abajo contiene:

1) 2px del borde del cuadro superior,
2) 30px de alto del cuadro,
3) 2px del filete inferior, y
4) 31px separado por el atributo margin-top.

Por lo tanto, podemos ver que el primer cuadro (**texto1**) toma como referencia la ventana (para ser más exactos **`<html>`** y **`<body>`** en los que está contenido), pero el segundo (**texto2**) toma como referencia el primero (**texto1**) y no la ventana, como sería de esperar.

Ubicación de elementos

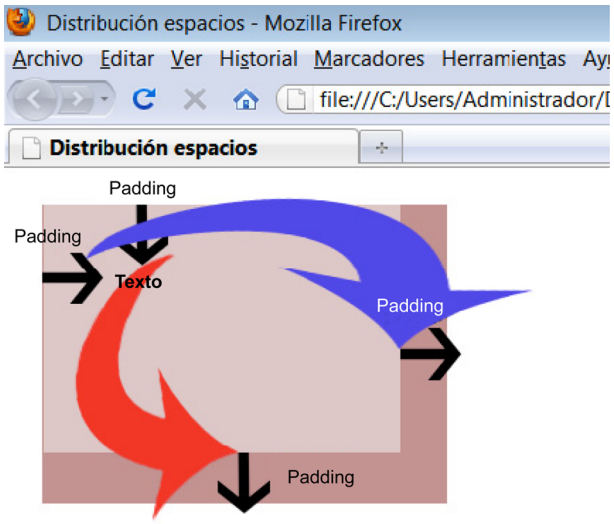
Considerar estos aspectos ahora puede parecer muy específico, pero va a ser el primer escollo que nos vamos a encontrar si pretendemos ser meticulosos con la ubicación de nuestros elementos.

Con este ejemplo debemos comprobar que los elementos están apilados unos sobre otros y que, de ese modo, los superiores se transforman en el límite de los inferiores.

Otro aspecto del mismo tipo que debemos observar en nuestro archivo HTML es el efecto del *padding* (recordemos que el *padding* es la distancia entre el contenido del cuadro y el filete o borde de éste). Intuitivamente no debería comportarse así, pero el cuadro se agranda tras la aplicación de padding-left y padding-top sobre el cuadro de texto2. Obsérvese que el ***padding*** **ha empujado los márgenes inferior y derecho del cuadro, con lo que ha hecho más grande a éste**.

Como veremos, posicionar dentro de una etiqueta con ***padding*** tiene un efecto inadecuado y muy poco intuitivo, aunque si se conoce no tiene por qué ser una molestia.

Tratamos aquí de representar esta situación mediante un gráfico:



Efecto "hinchado"

El modo más elemental de evitar el efecto de "hinchado" del *padding* será descontar de los laterales derecho e inferior de la etiqueta el *padding* aplicado en los atributos "width" y "height". O bien usar un posicionamiento negativo en los elementos que ocuparán esas posiciones.

3.5.4. Ya podemos colgar nuestro cuadro

Así, tenemos una idea de cómo actúa el posicionamiento de las etiquetas con **CSS**:

1) cada etiqueta se concibe como situada en un "cuadro o etiqueta" con fondo y figura;
2) las etiquetas están apiladas y se ubican por referencia a la etiqueta anterior;
3) margin define la parte exterior del cuadro;
4) en la parte interior del cuadro el efecto del *padding* hace crecer el rectángulo que sustenta la etiqueta.

Hay muchos más aspectos en el posicionamiento CSS, pero será fundamental que el diseñador neófito domine éstos para no volverse loco cuando las cajas no acaben de cuadrar en la maquetación. De ahí que hayamos incluido estas digresiones para explicarlo.

La intención de quienes se enfrentan por primera vez a estas prácticas de maquetación suele ser encajar todos los cuadros con exactitud. Estos aspectos que comentamos tratan de poner al estudiante sobre aviso de estos escollos iniciales, que por muchos motivos no se comentan en los manuales sistemáticos, aunque sí más a menudo en los manuales elaborados por diseñadores que en sus exposiciones van a problemas concretos (que son los más si examinamos la bibliografía).

Para lograr esta maquetación precisa y "milimétrica" (que ya veremos más tarde que siempre es recomendable) disponemos de un atributo en la mayoría de las etiquetas llamado ***position***, cuyos valores suelen ser ***relative*** o ***absolute***.

Asignando el valor absolute al atributo *position* podemos saltarnos este orden apilado convencional HTML, cambiar el flujo habitual de las etiquetas y su tendencia a apilarse y fluir a la derecha, y mediante este procedimiento hacer que los elementos tengan un posición fija, tal como haríamos sobre un papel, medida desde los lados izquierdo y superior de la ventana.

Posicionamiento absoluto

Sin embargo, no se recomienda el posicionamiento absoluto, pues no se ajusta a la multitud de los navegadores y las versiones de éste, o, por lo menos, ajustarlo para que se vea bien en la mayoría de los casos supondría un gran trabajo añadido.

Al tratarse de un documento estático, el navegador lo mostrará como una cámara que recorre una superficie mayor que su objetivo. Cuando, como sabemos, la idea principal es la contraria: que el diseño se pueda ver dentro de la ventana del navegador, sea ésta como sea y de manera completa, sin cortes, aunque a veces de ello resulte una composición un tanto peculiar.

Ved también

Más adelante, cuando tratemos la maquetación con (apartado 5), veremos cuál es el modo actual de hacer esto, pues la maquetación ha pasado por muchas fases que comentaremos en dicho apartado.

Otro aspecto importante es que Internet Explorer y Mozilla Firefox no interpretan igual ***margin*** y ***padding***; no obstante, aquí debemos tomar partido y entendemos que si se ve correctamente con Firefox es que cumple el estándar, y que Internet Explorer deberá adaptarse y dejar de intentar crear él estándar.

Webs recomendadas

W3C Recommendation. *6.13 Media descriptors*. http://www.w3.org/TR/1999/REC-html401-19991224/types.html#type-media-descriptors

José Alberto Torres Arroyo. "Introducción al segundo bloque del manual de tratamiento de imágenes con Javascript, que explica los objetivos que se deben perseguir". http://www.desarrolloweb.com/articulos/1167.php

W3C Recommendation. *12.3 Document relationships: the LINK element*: http://www.w3.org/TR/1999/REC-html401-19991224/struct/links.html#h-12.3

W3C Recommendation. *Font specification*. http://www.w3.org/TR/2008/REC-CSS2-20080411/fonts.html

W3C Recommendation. *Text specification*. http://www.w3.org/TR/2008/REC-CSS2-20080411/text.html

W3C Recommendation. *Cascading Style Sheets, level 2 CSS2 Specification*. http://www.w3.org/TR/2008/REC-CSS2-20080411/

Lecturas recomendadas

Autores varios (2008). *Profesional CSS para diseño Web*. Madrid: Anaya Multimedia.

J. Cranford Teague (2005). *DHTML y CSS*. Madrid: Anaya Multimedia.

Ch. Schmitt (2007). *Curso CSS*. Madrid: Anaya Multimedia.

R. York (2006). *CSS práctico*. Madrid: Anaya Multimedia.

4. Herramientas de diseño y navegación

4.1. Objetivos

El objetivo principal del presente apartado es que el estudiante conozca y pueda acceder a y elegir las herramientas de diseño HTML y navegación web más habituales actualmente.

Asimismo, el estudiante hará un repaso de las herramientas que tiene al alcance y, al final del apartado, tendrá suficientes conocimientos como para poder elegir tanto herramientas de diseño como de navegación que se ajusten a su situación, posibilidades y necesidades.

4.2. Editores

En relación con los editores webs y su codificación no resulta sencillo tener una idea clara de las ventajas y la funcionalidad que los diferentes editores pueden ofrecer. Lo que parece obvio es que hay que buscar el editor que se adapte mejor a las necesidades personales del usuario.

Entre los editores podemos diferenciar entre:

- Los **editores de texto/código fuente** son una aplicación especializada en facilitar la tarea de escribir código de diferentes lenguajes de programación (ejemplos: NOTEPAD, Bluefish, gedit).
- Los **editores visuales** WYSIWYG (Adobe Dreamweaver, Adobe GoLive, NVU/Kompozer, FrontPage, Amaya).

4.3. Editores WYSIWYG

Los editores WYSIWYG han sido a menudo criticados por añadir codificación a las normas, lo que produce un código sucio y sin cumplir los estándares que se han producido en los últimos años. No obstante, los editores WYSIWYG han mejorado y algunos producen un código válido y elegante. A veces es necesario utilizar herramientas sencillas para editar o actualizar los sitios web. Y aquí es donde la utilidad de los editores WYSIWYG puede resultar de utilidad.

¿Qué significa WYSIWYG?

La abreviatura de alguna manera críptica WYSIWYG significa *what you see is what you get* (Lo que ves es lo que obtienes).

En estos editores no hay que editar directamente el código fuente HTML de los documentos, pero su presentación aparecerá en el documento final.

En el caso de los editores de HTML, este concepto se aplica a aquellos programas que permiten escribir la página sobre una vista preliminar parecida a la de un procesador de textos, ocupándose el programa, en este caso, de generar el

código fuente en HTML. Así, en lugar de escribir bloques de código de manera manual (como se haría con el Bloque de Notas o LaTeX), se manipulan con los componentes de diseño utilizando una ventana de editor. Eso significa que mientras se crea el documento se ve una cosa muy similar al resultado final, mientras que el documento o la imagen se están creando.

4.4. Control del código de diseño

La mayoría de los editores WYSIWYG presentan la posibilidad de cambio de visualización de vistas (vista código HTML, vista compartida y vista diseño). El modo Diseño muestra un resultado parecido al resultado final, pero no exacto; se debe ver la presentación previa o presentación final con los navegadores para comprobar el resultado final una vez la página esté disponible en Internet.

La **vista Diseño** muestra sólo el texto principal de un documento, la parte situada entre el **`<body>`** y las etiquetas **`</body>`**. Aunque se pueden editar algunas propiedades del elemento **`<head>`**, como el título del documento, mediante la ventana **Propiedades** de éste, debe alternar la vista Código Fuente para editar las propiedades de los elementos que no están en el elemento **`<body>`**.

4.4.1. Diferencias entre la vista Diseño y los navegadores web

La vista Diseño de los editores web proporciona una vista de edición que se aproxima bastante al modo WYSIWYG del aspecto que tendrá la página en un navegador. No obstante, esta vista no es una coincidencia de la representación de la página y siempre se debe probar la página en un navegador (o en varios) para estar seguro de que la página aparecerá cuando se ha diseñado.

Un documento mostrado en la vista Diseño se diferencia de otro mostrado en un navegador web en lo siguiente:

1) La superficie de diseño se puede modificar.
2) Algunos elementos se muestran en la vista Diseño con la finalidad de editarlos, pero no se representan en el navegador, como los controles de origen de datos.
3) Algunos formatos de caracteres y de párrafo pueden mostrarse de manera diferente en un navegador específico (si el navegador web implementa el formato de manera diferente a la vista Diseño).
4) Opcionalmente, puede tener los cuadros de presentación del editor, los símbolos y los iconos para marcar los controles de servidor.
5) Los hipervínculos no funcionan.
6) Las secuencias de pedidos de cliente no se ejecutan.
7) El código del servidor no se ejecuta.

8) Los elementos que admiten texto alternativo (como las imágenes) no lo muestran como información sobre herramientas al pasar el puntero sobre ellos.

Muestra elementos que no se ven

Para ayudar a editar una página, la vista Diseño muestra algunos elementos, como los campos ocultos, que normalmente no se verían en el navegador.

Además, puede decidir mostrar los bordes y los símbolos que pueden ayudar a trabajar con elementos y etiquetas en la página.

Colocar elementos en la vista Diseño

Los elementos de la página se diseñan físicamente de arriba abajo. Por defecto, cuando la página se representa en el navegador, los elementos se representan en el mismo orden de arriba abajo. Asimismo, puede diseñar los elementos bidimensionales, poniéndolos mediante coordenadas horizontales y verticales en cualquier parte de la página. Esta opción de diseño aprovecha las opciones de posición disponibles mediante los estilos.

Explorar la vista Diseño

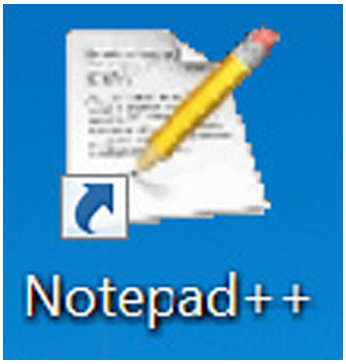
Para ayudar a desplazarse por los elementos y seleccionarlos, la vista Diseño de algunos editores proporciona algunas de estas opciones:

- **Navegador de etiquetas**: el navegador de etiquetas muestra el elemento actual, junto a la jerarquía de etiquetas primarias a las que pertenece. Podéis utilizarlo para ver el elemento que tiene el foco y para desplazarse desde el elemento actual a un elemento superior de la jerarquía.
- **Esquema del documento**: la ventana Esquema del documento permite buscar y seleccionar todos los elementos de un documento, incluidos los que no se muestran
- **Ventana Propiedades**: cuando se selecciona un elemento de la lista desplegable situada en la parte superior de la ventana Propiedades, el editor selecciona este elemento del documento.

4.5. Herramientas

4.5.1. Herramientas básicas: NOTEPAD

El **Notepad++** es un editor de código fuente libre que soporta varios lenguajes de programación y funciona en el entorno MS Windows y bajo licencia GPL.

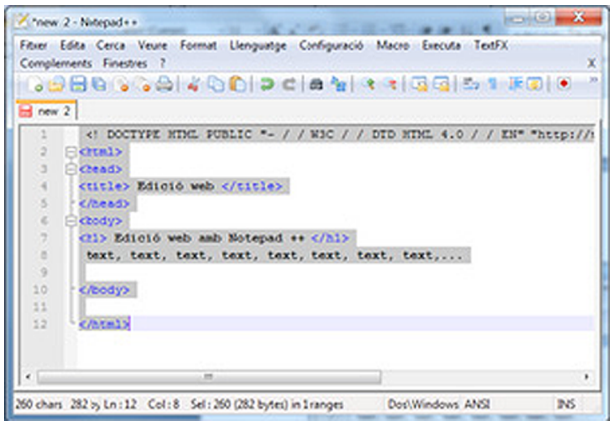


Icono Notepad++

Características del Notepad++:

- Sintaxis con colores y envoltura de sintaxis. Permite imprimir el código fuente en color.
- Autocompletado de etiquetas de los diferentes lenguajes de programación que soporta.
- WYSIWYG (lo que ves es lo que obtienes).
- Multi-Documento y Multi-Vista. El programa permite editar diferentes documentos al mismo tiempo, así como visualizarlos también a la vez. La modificación del documento en una vista quedará reflejada también en la otra, es decir, se modifica el mismo documento, aunque en realidad se ve una copia. El usuario puede definir la posición de las vistas dinámicamente; el separador puede ser tanto horizontal como vertical.
- Soporte de busca y sustitución con expresiones regulares.
- Soporte completo de arrastrar y soltar: permite abrir y mover desde una posición, o incluso de una vista, a otra un documento arrastrándolo y soltándolo.
- Autodetección del estado del fichero. Si se modifica o suprime un fichero abierto con el Notepad++, el propio programa lo notifica, y o bien da la opción de volver a cargarlo o bien de suprimirlo.
- Acercar y alejar el texto es otra función del componente Scintilla.

Captura Notepad++



4.5.2. Herramientas *freeware*: NVU/KOMPOSER, Amaya

El **NVU** (pronunciado en inglés como *N-view*) es un completo sistema para editar páginas web que combina la gestión de todos los ficheros del sitio web y la facilidad de uso del WYSIWYG.

El NVU es un editor basado en Mozilla Composer, pero de ejecución independiente.

El KompoZer es una derivación de proyecto NVU (actualmente abandonado) con diferentes errores corregidos.

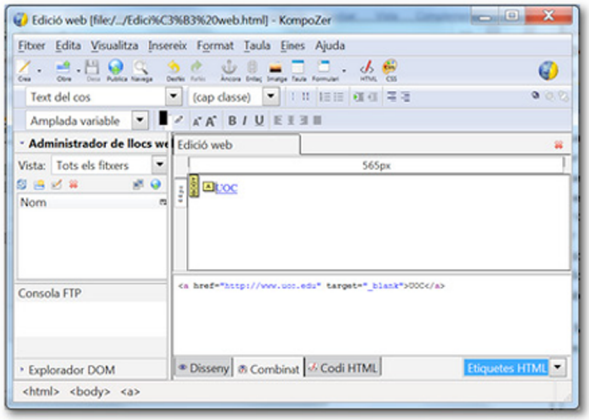
KompoZer es un completo sistema de autoría web que combina la gestión de archivos web y de edición WYSIWYG.

KompoZer está diseñado para ser muy fácil de usar, por lo que es una solución viable para los usuarios que quieren crear sitios web sin la obtención de conocimientos técnicos. En comparación con NVU, presenta un limpiador de marcado y tiene marcas visibles. La última versión se publicó en agosto del 2008. KompoZer se puede considerar una buena alternativa para pequeños proyectos y para los usuarios sin conocimientos técnicos. El programa añade características como soporte integrado de CSS y mejor gestión de FTP para la actualización de ficheros. Pero quizá es débil para el desarrollo web profesional.

Icono Komposer

El programa está disponible para GNU/Linux, Mac OSX y Microsoft Windows, e incluso existe una versión portátil (o "portable") que puede ser utilizada desde una memoria USB sin necesidad de instalarla en el ordenador.

Captura pantalla de edición del Komposer



Amaya es una herramienta combinada del W3C compuesta por un navegador web y una herramienta de autor.

Cualquier página web que se abra puede ser editada inmediatamente. Permite ver y generar páginas HTML y XHTML con hojas de estilo CSS, etc.

Es capaz de trabajar en diferentes documentos al mismo tiempo (varios (X) HTML, MathML nativa (.MML) y SVG (.SVG), que se pueden visualizar y editar a la vez).

Amaya también incluye una aplicación de anotación colaborativa (anotaciones son comentarios externos, notas, comentarios que se pueden conectar a cualquier documento web o a una parte seleccionada del documento) y que tiene soporte para SVG, RDF y XPointer.Open-Source.

4.5.3. Herramientas de autor: DREAMWEAVER

Adobe Dreamweaver, originalmente creado por Macromedia, es una herramienta profesional de desarrollo de proyectos web.

Las versiones iniciales de la aplicación servían de editores de HTML WYSIWYG simples, pero versiones más recientes han incorporado un notable soporte para muchas otras tecnologías, como CSS y JavaScript, entre otros.

El software está disponible para las plataformas de Mac y Windows, pero también puede ser utilizado en sistemas operativos basados en Unix, siempre que funcione mediante una emulación virtual de la plataforma del MS Windows. Desde el año 2005 el Macromedia Dreamweaver es propiedad de Adobe Systems.

Icono del Dreamweaver

Como cualquier editor de WYSIWYG, Adobe Dreamweaver permite ocultar los detalles del código de HTML de las páginas del usuario, lo que posibilita que los diseñadores web desconocedores del lenguaje HTML puedan crear páginas web con la técnica de arrastrar y soltar (*drag and drop*).

Sin embargo, en las últimas versiones de la aplicación se ha aumentado el soporte para CSS y propone el trabajo con capas y *divs*.

Adobe Dreamweaver, anteriormente Macromedia Dreamweaver, permite a los usuarios visionar los sitios web utilizando otros navegadores, con la única condición de que se instalen en el ordenador. También incorpora herramientas de

gestión de sitios, así como la posibilidad para encontrar y reemplazar líneas de texto, entre otras opciones. También permite el uso de JavaScript básico sin ningún conocimiento de codificación.

Es uno de los editores de uso común que pueden dar soporte a los desarrolladores, mejorar el flujo de trabajo y ahorrar bastante tiempo durante la codificación. En las últimas versiones el código de marcas es muy limpio.

Dreamweaver también ofrece numerosas herramientas útiles, como la biblioteca de fragmentos de código, la gestión de FTP, el servidor de depuración y un desarrollo integrado de codificación. Por ejemplo, se puede ver información de la CSS en un único panel CSS unificado, que facilita ver los estilos aplicados a un elemento específico, identificar dónde se definen los atributos y modificar estilos existentes, sin entrar en la vista Código.

4.6. Los navegadores

Un **navegador web** es una aplicación que permite al usuario recuperar y reproducir documentos de hipertexto, generalmente escritos en HTML, desde servidores web situados en cualquier lugar del mundo.

Esta red de documentos se conoce como World Wide Web (WWW).

Los navegadores actuales

Los navegadores actuales permiten mostrar gráficos, secuencias de vídeo, sonidos, animaciones y programas diferentes, además de texto y enlaces.

La función básica de un navegador web es mostrar documentos de texto, seguramente con recursos multimedia incrustados. Los documentos pueden estar ubicados en el ordenador del usuario, pero también pueden estar en cualquier otro dispositivo que esté conectado a él (a través de Internet, por ejemplo) y que tenga los recursos necesarios para la transmisión (un software de servidor web). Estos documentos, habitualmente denominados **páginas web**, tienen hipervínculos que enlazan una porción de texto o una imagen con otro documento.

El seguimiento de enlaces de una página a otra, ubicada en cualquier ordenador conectado a Internet, se conoce como navegación.

4.6.1. Internet Explorer

Windows **Internet Explorer** (*IE* o ***MSIE***) es un navegador propietario desarrollado por Microsoft y que se distribuye conjuntamente a partir de la versión 98 de Microsoft Windows.

Internet Explorer es uno de los navegadores web más utilizados, aunque ha perdido una parte del mercado, en parte por los problemas de seguridad y en parte por la aparición de navegadores libres como Firefox.

Versiones de Internet Explorer

- **Internet Explorer 6**

Apareció en el año 2001, incluyendo características del DHTML, restricción de marcos interiores para el contenido y soporte parcial para el CSS, entre otras características. También aportaba una nueva versión del Kit de Administración de Internet Explorer, la barra de medios, integración de la mensajería instantánea de Windows, Messenger y un redimensionado automático de las imágenes; el P3P, un protocolo para la protección de la privacidad en la Red y una nueva apariencia, más en la línea visual de Windows XP.

- **Internet Explorer 7**

En febrero del 2005, el presidente de Microsoft, Bill Gates, anunció la aparición de la nueva versión del navegador. La decisión de actualizar el navegador se tomó después de la expansión de Mozilla Firefox. La versión definitiva se hizo pública en octubre del 2006. Esta versión 7 de Internet Explorer sólo está disponible para Windows XP y versiones posteriores.

Esta versión 7 intenta defender a los usuarios del *phishing* o captura de datos personales, así como del software engañoso o delictivo; también otorga control absoluto al usuario del *ActiveX* y un mejor marco, lo que incrementa la seguridad en la navegación.

Incluye, finalmente, soporte de pestañas como el resto de navegadores actuales.

- **Internet Explorer 8**

Internet Explorer 8 es la octava versión publicada del navegador web de Microsoft y sucede a Internet Explorer 7. Fue lanzado el 19 de marzo del 2009 como actualización para Windows XP Service Pack 2 o superior, Windows Server 2003 con Service Pack 1 o posterior, Windows Vista y Windows Server 2008. Internet Explorer 8 se incluirá de manera nativa en los próximos sistemas operativos de Microsoft, Windows 7 y Windows Server 2008 R2.

Icono Internet Explorer

Según Microsoft, las prioridades del IE8 son la seguridad, la facilidad de uso, las mejoras de RSS, CSS y el soporte en tecnologías basadas en AJAX, con una considerable mejora del soporte en los estándares web con respecto a su precursor.

4.6.2. Firefox

Mozilla Firefox es un navegador web libre desarrollado a partir del proyecto Mozilla.

El objetivo de Firefox es desarrollar un Mozilla más pequeño, ligero y rápido mediante la extracción y el rediseño del componente de navegador de la *suite* de software Mozilla. Al igual que Mozilla, Firefox es multiplataforma.

Icono Mozilla Firefox

Mozilla Firefox es un proyecto paralelo al *Mozilla Thunderbird*, un cliente de correo electrónico bajo la misma filosofía y con unos orígenes similares. El buen recibimiento de ambos por parte de los usuarios llevó a que la Fundación Mozilla los adoptara como uno sus productos principales, sustituyendo a la *suite* Mozilla original.

Algunas de las principales características de Mozilla Firefox son:

- **Navegación con pestañas**: Firefox abre las nuevas páginas en nuevas pestañas por defecto. Cuando hay demasiadas pestañas abiertas, aparecen barras de desplazamiento en cada lado y un botón a la derecha que muestra una lista, fácil de leer, de todas las que están abiertas.
- **Restauración de la sesión**: esta funcionalidad permite recuperar las ventanas, pestañas, texto escrito en los formularios y las descargas en curso de la última sesión de usuario, si el Firefox se debe reiniciar o cerrar.
- **Motores de búsqueda**: incorpora algunos motores de búsqueda, como Google, Yahoo!, el Gran diccionario de la lengua catalana o la Wikipedia, que pueden sugerir términos de busca; sólo hay que empezar a escribir. En cualquier momento, se puede seleccionar un nuevo motor del menú o añadirlo. Firefox notifica cuándo algún sitio web ofrece nuevos motores de búsqueda que se puedan instalar y los instala con un clic.
- **Canales de información web**: Firefox da control total sobre los canales de información, dejando elegir cómo se quiere la suscripción.
- **Direcciones de interés activas**: las direcciones de interés activas del Firefox permiten ver canales de información web como noticias o bloques en la barra o el menú de las direcciones de interés.
- **Bloqueo de ventanas emergentes**: el usuario tiene el control de las páginas web visitadas mediante el bloqueo de ventanas emergentes. Asimismo, notifica cuándo una ventana emergente ha sido bloqueada.
- **Actualización automática**: Firefox comprueba siempre si se está utilizado la última versión y avisa cuando existe disponible una actualización de seguridad.
- **Cambio visual**: el tema de Firefox y la interfaz de usuario se han actualizado para mejorar la usabilidad sin alterar la familiaridad en la navegación.
- **Un complemento para todo el mundo**: se puede elegir entre más de 1.000 complementos en http://www.addons.mozilla.com.

- **Gestor de complementos para extensiones y temas**: mejora la interfaz de usuario para gestionar las extensiones y los temas, haciendo más sencilla la personalización de Firefox.

4.6.3. Opera

Opera es un navegador de Internet creado por la empresa noruega Opera Software en 1994.

Opera es gratuito desde la versión 8.50. El navegador Opera es conocido por su velocidad, seguridad, soporte a los estándares (especialmente CSS), tamaño reducido, internacionalidad y constante innovación. Fue uno de los primeros navegadores en implementar la navegación de sitios web por pestañas.

Se pueden encontrar distribuciones de este navegador para plataformas como Windows, Apple Macintosh y Linux, pero se encuentra también disponible para otras plataformas. Además, existen versiones compactas para una amplia variedad de dispositivos móviles.

Icono Opera

Actualmente se encuentra en su versión 10.10, que destaca por las prestaciones que ofrece:

- **Sistema de pestañas**: permite tener diferentes webs en diferentes pestañas.
- **Guarda las sesiones**.
- **Recuperación de páginas cerradas**: permite la recuperación de páginas ya cerradas.
- **Lector de RSS**: correo electrónico, noticias, sindicaciones, etc.
- **Chat** integrado.
- **Sistema *antiphishing*** que nos mantendrá alerta en el caso de acceder a páginas poco recomendables.
- **Motores de búsqueda**: posibilidad de añadir motores de búsqueda. Sólo es necesario ir al campo de busca de una web y hacer clic en el botón derecho para seleccionar "Crear búsqueda".
- **Bloqueo de contenidos**: donde se puede indicar ver o no imágenes, *banners*, etc., en determinados sitios web.
- **Edición de las preferencias**, para cada sitio web, ventanas emergentes, *cookies*, bloqueo de imágenes o identificación del navegador.
- ***Widgets* o complementos**, equivalentes a los complementos del navegador Mozilla Firefox, que añaden nuevas funcionalidades al navegador Opera.
- **Edición avanzada de texto**.
- **Vista previa** de los contenidos de cada pestaña. Basta con ponerse encima de la pestaña para que se abra una pequeña ventana emergente.

4.6.4. Safari

Safari es un navegador web desarrollado por Apple para su sistema operativo Mac OS X.

La versión 3.02 está disponible para Windows XP y Windows Vista. El código utilizado está basado en el proyecto KDE. El motor interno de Safari es un software libre liberado bajo los términos de la licencia GPL.

Icono Safari

Se trata de un navegador simple y muy rápido. Safari es el doble de rápido al iniciar y cargar páginas, si se compara con Internet Explorer, y 1,6 veces más rápido que el navegador de código abierto Firefox. Tiene una interfaz gráfica sencilla, que no distrae y permite enfocarse plenamente en el lugar en el que se está navegando.

Las características de este navegador son:

- Motor de renderizado KHTML.
- Interfaz de estilo metálico.
- Administración de marcadores sencilla.
- Permite la seguridad y la navegación privada, así como el bloqueo de ventanas emergentes.
- Integración de la tecnología multimedia de Apple QuickTime.
- Navegación dividida en pestañas, al igual que la mayoría de los navegadores actuales.
- Caja de texto para buscas, por medio del motor de búsqueda de Google, además de una busca en línea.
- Lectura de canales de información, *feeds* (RSS).
- Guarda páginas en archivos locales.
- El *SnapBack* permite volver a los resultados de la búsqueda original o al primer nivel de cualquier página web, aunque esté a unos cuantos niveles de distancia.

4.6.5. Google Chrome

Google Chrome es un navegador web desarrollado por Google y compilado con base en componentes de código abierto como el motor de renderizado de WebKit y su estructura de desarrollo de aplicaciones (Framework).

Google Chrome es el tercer navegador más utilizado.

Icono Google Chrome

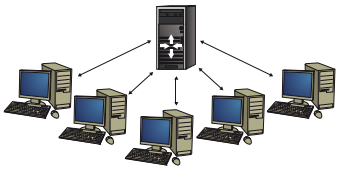
Está disponible gratuitamente bajo condiciones de servicio específicas. El nombre del navegador deriva del término utilizado para el marco de la interfaz gráfica de usuario ("chrome").

Chromium es el proyecto de software libre detrás de Chrome. Su objetivo principal es proporcionar un navegador con mayor estabilidad, velocidad y seguridad, además de incluir una interfaz de usuario sencilla y eficiente. En esencia, Chromium es el navegador base desde el que está construido Chrome y tiene sus mismas características de diseño, pero con un logotipo ligeramente diferente y sin el soporte comercial y técnico de la compañía Google.

El 2 de septiembre del 2008 salió a la luz la primera versión al mercado y actualmente se encuentra en la versión 4. De momento, el navegador está disponible para la plataforma Microsoft Windows en más de 50 idiomas. La versión para sistemas Mac OS X y Linux se encuentra actualmente en desarrollo, con disponibilidad de versiones beta en los dos sistemas operativos.

4.7. Arquitectura cliente-servidor

La web es uno de los ejemplos más representativos de lo que se entiende como relación cliente-servidor en una arquitectura informática.


Esquema de cliente-servidor

En la relación cliente-servidor, el cliente (que suele ser una aplicación que emplea una interfaz gráfica) realiza peticiones al servidor para que le devuelva la información solicitada. En esta relación, el cliente es el agente activo (es él quien inicia la relación al enviar preguntas al servidor y esperar las respuestas), mientras que el servidor es el elemento pasivo que espera ser preguntado.

Existen diferentes tipos de servidores, en función de cuál sea su propósito, aunque su arquitectura sea la misma. Veamos algunos de estos servidores:

- **servidores web** (que aceptan las preguntas http de los clientes y que da respuestas que, usualmente, son páginas web en formato html);
- **servidores de aplicaciones** (es un motor de software dedicado a poner en funcionamiento ciertas aplicaciones);
- **servidores terminales** (sirven para conectar múltiples sistemas, posiblemente de manera remota hacia un procesador central) y
- **los servidores de correo** (también conocido como MTA, *mail transfer agent*, que en el contexto de los sistemas de nombres de dominio es un software que transfiere los mensajes de correo electrónico de un ordenador a otro).

Los servidores pueden ser *stateless* o *stateful*. Los servidores *stateless* tratan cada pregunta de manera independiente y, por lo tanto, no mantienen información entre preguntas ni las relacionan. Un ejemplo de este tipo de servidor puede ser las páginas http estáticas. Este tipo de servidor tiene la ventaja de su diseño simple, pero la desventaja de necesitar más información en cada pregunta.

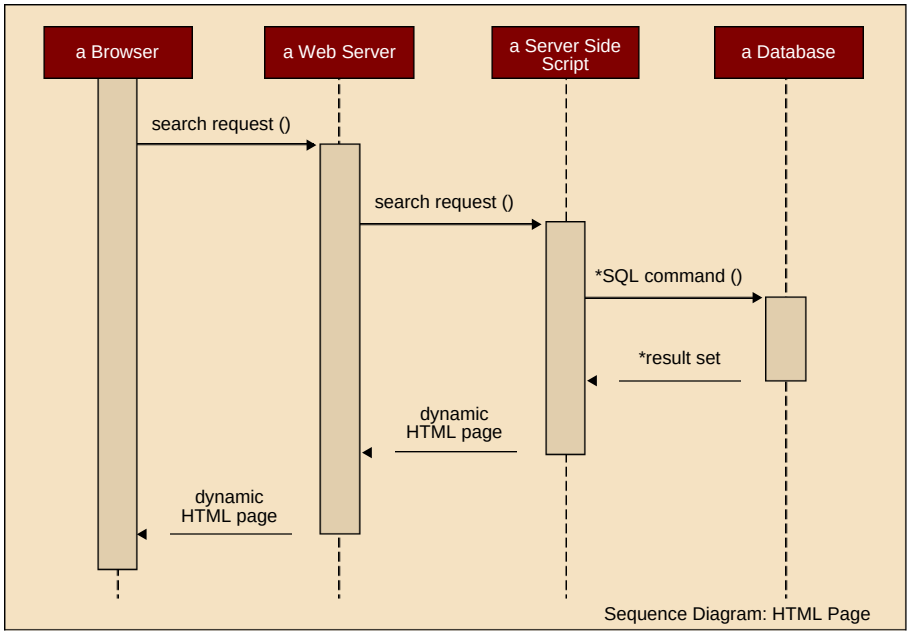
Por su parte, un servidor *stateful* mantiene la información de las preguntas anteriores y las relaciona, con lo que proporciona unas respuestas más acertadas ante preguntas o demandas más simples, eso sí, con la necesidad de un diseño mucho más complejo.

Ejemplo

Como ejemplo tenemos el servidor Apache Tomcat, que utiliza el lenguaje Java.

Muy a menudo, la relación cliente/servidor es descrita como una secuencia de diagramas que nos muestra, con dos líneas verticales paralelas, diferentes procesos u objetos que interactúan simultáneamente y que, con flechas horizontales, marca los mensajes que se envían en el orden en que se han dado. Esta secuencia está estandarizada por UML (*unified modeling language*).

Diagrama cliente-servidor



Este tipo de arquitectura, cliente/servidor, es conocida habitualmente como arquitectura de dos niveles, aunque también se puede hablar de una arquitectura de tres niveles formada por el cliente, los servidores de aplicaciones y los servidores de las bases de datos que almacenan la información para los anteriores, e incluso de una arquitectura de *n*-niveles con más de dos servidores para facilitar los resultados, mucho más difícil de programar.

4.8. La vida de una página web

Cuando se crea una página web hay que enviarla a un servidor web para que pueda ser visualizada en la Red.

Un servidor web es un programa que sirve datos en forma de páginas web, hipertextos o páginas HTML (*hypertext markup language*): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados, como animaciones o reproductores de sonidos.

La comunicación de estos datos entre cliente y servidor se realiza mediante un protocolo, concretamente el **protocolo HTTP**.

Protocolo

Conjunto de reglas que gobiernan el intercambio de datos entre entidades dentro de una red. Es el lenguaje común "que utilizan" los ordenadores para "hablar" y entenderse entre sí.

Hay muchos tipos de protocolos, cada uno con sus reglas bien definidas, como FTP, POP3, SMTP, ICMP, etc.

Así, un servidor web se mantiene a la espera de peticiones HTTP, que son ejecutadas por un cliente HTTP, que solemos conocer como navegador web.

A modo de ejemplo: al teclear http://www.uoc.edu en un navegador, éste realizará una petición HTTP al servidor que tiene asociada esta URL. El servidor responderá al cliente enviando el código HTML de la página y el navegador, al recibir el código, lo interpretará y mostrará en la pantalla.

El cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página. El servidor se encarga de entregar el código de la página sin realizar ninguna interpretación de ésta.

Protocolo HTTP

Una de las características del protocolo HTTP es que no es permanente, es decir, cada operación HTTP implica una conexión con el servidor, que es liberada cuando termina.

Ejemplo

Por ejemplo, para un documento HTML con 10 imágenes son necesarias 11 conexiones diferentes (10 imágenes más la página HTML en sí).

Además, no tiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.

A partir de la versión 1.1 del protocolo HTTP, se pueden habilitar conexiones persistentes (permiten enviar más objetos con un menor número de conexiones).

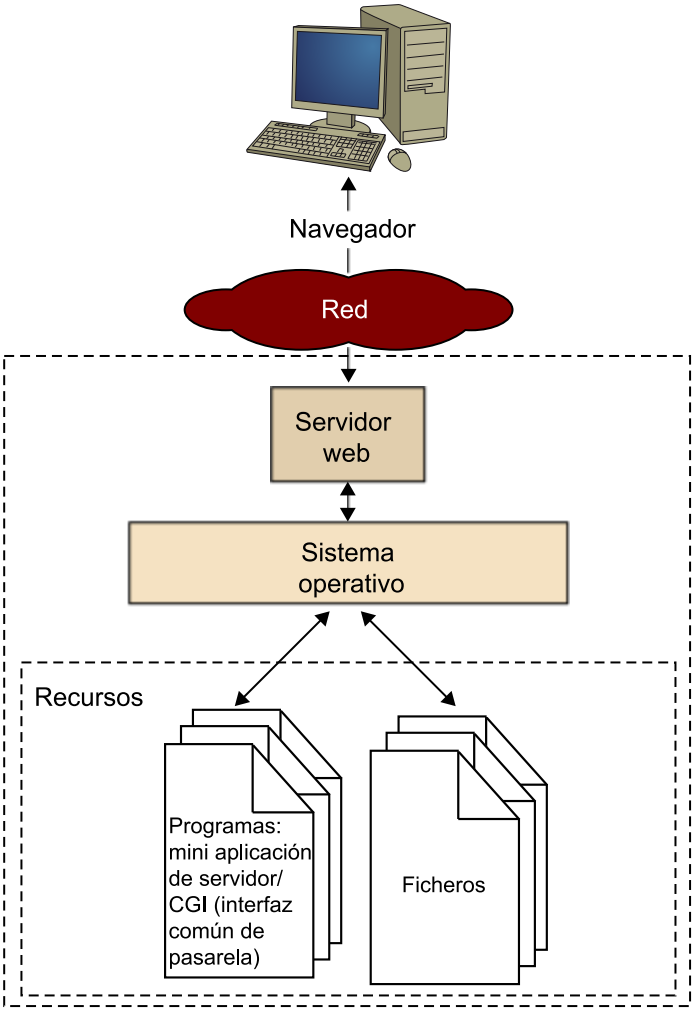
La petición de una página web se realiza en dos pasos:

1) El navegador pide como cliente DNS la traducción de una URL (por ejemplo, http://www.uoc.edu) a una IP, y

2) una vez que ha recibido la traducción del servidor DNS, se realiza la petición HTTP al servidor que tenga la IP concreta.

Aunque los diferentes programas servidores web difieren en los detalles, todos comparten unas funcionalidades básicas.

Interacción servidor y entorno



La imagen muestra la interacción entre un servidor web y el resto del entorno. El servidor es el responsable de proporcionar el acceso a los recursos solicitados que están bajo el control del sistema operativo. Estos recursos pueden ser:

- **estáticos**, como páginas HTML o texto, y
- **dinámicos**, programas que son ejecutados por el servidor. Digamos que es la parte inteligente del servidor.

Lecturas recomendadas

A. Martos (2009). *Internet*. Madrid: Anaya, DL. ISBN 9788441525801.

G. McGovern; R. Norton; C. O'dowd (2002). *The Web Content Style Guide. And essential reference for online writers, editors and managers*. Gran Bretaña: Pearson Education. ISBN: 0273656058.

J. Restrepo (2001). *Internet para todos*. Random House Español. ISBN: ISBN0375719652, 9780375719653.

Webs recomendadas

Navegadores.org. http://navegadores.org/ [en línea] [Última visita 23/04/2010]

Mozilla hispano. http://www.mozilla-hispano.org/ en línea] [Última visita 23/04/2010]

Maestros del web. http://www.maestrosdelweb.com/ [en línea] [Última visita 23/04/2010]

Editores web. http://www.cdlibre.org/consultar/catalogo/Desarrollo-Web_Editores-web.html [en línea] [Última visita 23/04/2010]

Las novedades de HTML 5. http://www.anexom.es/tecnologia/diseno-web/las-novedades-de-html5-i/ [en línea] [Última visita 23/04/2010]

5. Maquetación de un documento con CSS

5.1. Objetivos

El objetivo de este apartado es que el estudiante tenga una visión general de lo que ha sido la maquetación desde los comienzos del HTML, su progreso y sus distintas opciones. Que aprenda a valorar los pros y los contras de las distintas opciones que se le presentan, tablas, marcos y CSS, y que esto le permita elegir lo que se ajuste mejor a su situación, posibilidades y necesidades.

5.2. Introducción

Ya sabemos dar aspecto visual a una etiqueta y posicionarla con **CSS**, aunque sea de manera sencilla. Pero ¿cómo empleamos todo esto? ¿Por dónde empezar o qué hacer?

Bien, presuponemos que el estudiante ya ha leído los apartados anteriores referidos al **HTML** y que está al tanto de los rudimentos de programación con un editor de texto para crear HTML. Además, que ha leído el apartado referido a las hojas de estilo en cascada (HEC) CSS y que sabe cómo mantener la conexión entre el documento HTML y el CSS mediante **`<link>`**. Con todo dispuesto comienza a crear su primer documento basado en hojas de estilo en cascada (HEC).

5.3. Un poco de historia

En resumen, cabe tener en cuenta que en las primeras implementaciones de HTML no existía aún la idea de hojas de estilo en cascada (HEC); por lo tanto, los documentos resultaban en un principio muy operativos pero poco vistosos.

En el fondo, el texto no necesita mucho adorno para ser leído y entendido, así que un texto era un montón de etiquetas apiladas unas bajo otras, de modo que entre títulos (**`<h1>`**...), párrafos (**`<p>`**) e imágenes (**`<img>`**) discurría la mayor parte del documento, adornado como mucho por alguna lista (**`<ol>`** y **`<ul>`**) y salpicado de enlaces; como decoración se solía utilizar la negrita **`<b>`**, la cursiva **`<i>`** y el subrayado **`<u>`**.

Nota

Recuérdese que hoy **`<b>`** es **`<strong>`** y que **`<i>`** se ha convertido en **`<em>`**.

Lo cierto es que para un documento de lectura esto sigue, a día de hoy, siendo más que suficiente, pues responde al concepto de documento heredado de la cultura de la impresión del libro. Pero la web se transmitía mediante el monitor de un ordenador y la televisión había estado ocupando ese espacio de la pantalla durante muchos años.

No pretendemos aquí determinar las causas, pero la parte visual del monitor reclamaba su espacio. ¿Cómo implementar con nuestro HTML y CSS todo el aparato visual de las revistas –también material impreso– y de los restantes medios audiovisuales como la televisión, el vídeo y ya, por entonces, el CDROM?

No sólo había que apilar textos como en un principio, sino que se hizo necesario ordenar los textos en pantalla y maquetar la página de manera que tuviera un aspecto visual más agradable y más adaptado a lo que el público en general estaba acostumbrado a percibir.

Este proceso pasó por tres fases que se han ido descartando en el mundo del diseño, pero que, hasta cierto punto, todavía están vigentes si alguien pretende construir un documento sencillo.

En un primer momento se comenzó a maquetar a partir de tablas, usando **`<table>`** y sus posibilidades para hacer la distribución estructural del documento, luego se ocultaban los bordes de las tablas y se colocaba cada texto en su lugar.

Es sencillo y además, aunque a día de hoy no se usa en la concepción global de la maquetación, se usa muchísimo a pequeña escala, en las partes más internas del documento.

Maquetación a partir de `<div>`

Algunos CMS como PHPNUKE, que suponen el último paso en la programación web, maquetan a partir de divisiones (**`<div>`**), pero dentro de las divisiones se usan muchísimo las tablas.

No obstante, las tablas resultaron muy rígidas. Hay que tener en cuenta que un concepto fundamental que se iba imponiendo en la Web es que el contenido, sea éste el que sea, pueda ser visto y/o distribuido de diferentes maneras, dada la multiplicidad de pantallas en las que la hoja web iba a ser visitada. Y la rigidez mostrada por las tablas, incluso trabajando con planteamientos escalares, hizo aconsejable otra aproximación.

Otro aspecto fundamental para descartar las tablas resultó la accesibilidad, pues los lectores para invidentes encuentran muy difícil circular por las tablas manteniendo una lectura del texto coherente.

La maquetación también se puede realizar a partir de marcos **`<frame>`**. Lo primero que quiere hacer un diseñador novel es tener un menú a la izquierda con enlaces que vayan apareciendo a la derecha. Esta estructura de índice reproduce el modo más habitual de manejar las publicaciones impresas y resulta sumamente intuitiva.

Pese a ser el manejo de los marcos interesante y práctico, éstos están muy denostados y se aducen muchas razones por las que no debieran ser utilizados:

1) confunden a las arañas de búsqueda;
2) aumentan los tiempos de carga;
3) impiden la numeración correcta de páginas;
4) impiden una correcta incorporación de un marcador a la página;

5) realizar versiones con marcos y sin ellos de la misma página tiene un coste añadido (y hay que hacerlo si queremos ser accedidos por terminales que no cuentan con marcos o que tienen el área de visión pequeña); y
6) el cierre de la web o salida puede verse impedido, si el entramado de marcos no es claro y funcional.

Se puede seguir este debate a partir de multitud de enlaces en GOOGLE.

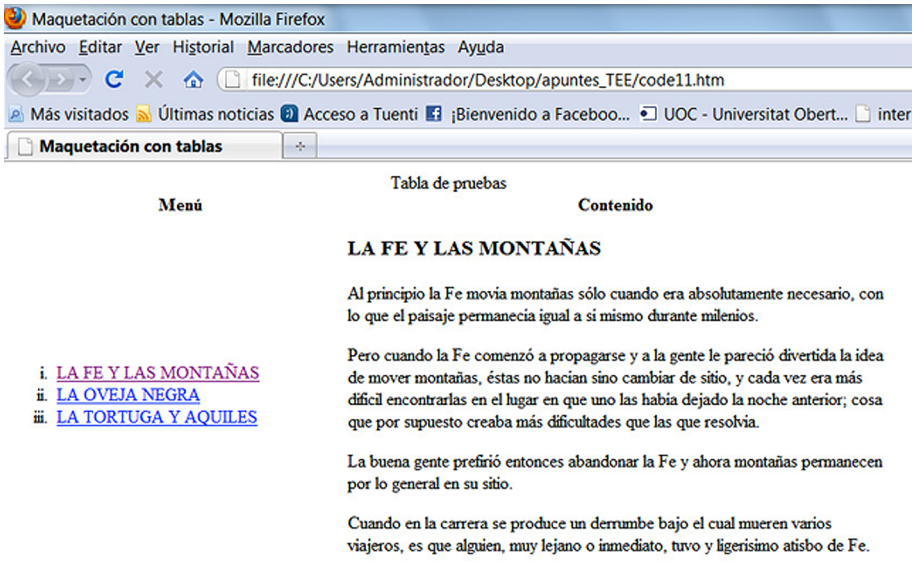
No obstante, unas cosas y otras tienen sus desventajas, aunque tienen también sus ventajas. Por ello, hablaremos aquí de las tablas y de los marcos como introducción a la comprensión posterior de **`<div>`** y su utilización para la navegación y maquetación.

5.4. Maquetación con tablas (`<table>`)

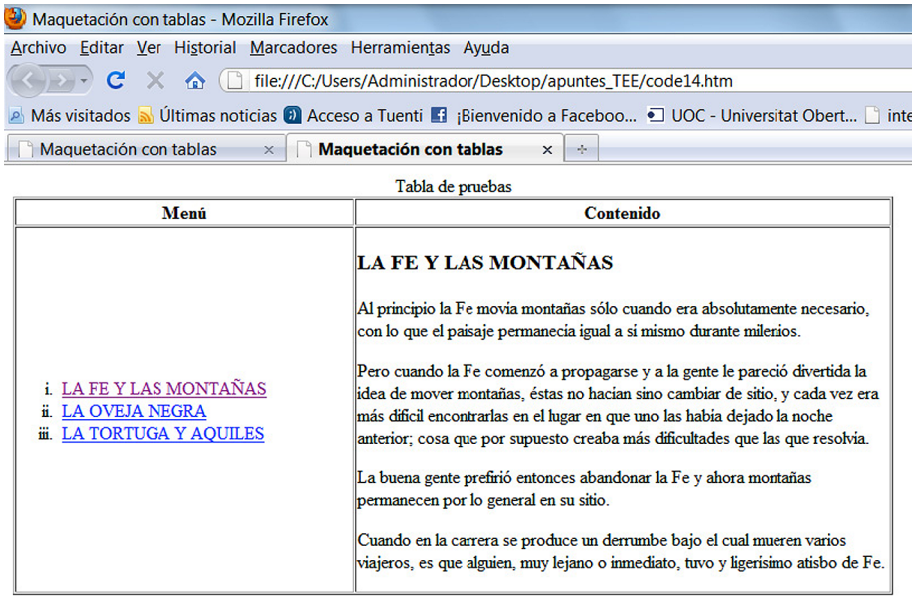
Presuponemos que se conoce el **HTML** básico que permite manejar una tabla **`<table>`**. De no ser así, recomendamos revisar las etiquetas que hacen referencia a este ámbito **`<table>`**, **`<caption>`**, **`<th>`**, **`<td>`**, **`<tr>`**, etc.

No resulta difícil imaginar que colocando dentro de cada celda de la tabla un fragmento de texto se pueda obtener un documento con el aspecto habitual de menú a la izquierda y contenido a la derecha.

Observemos el funcionamiento de los ejemplos.



Si actuamos sobre el menú de la izquierda en el archivo de trabajo, veremos cómo el menú hace su trabajo y el texto mantiene sus posiciones y estructura porque la tabla de base sostiene todas y cada una de las páginas que intervienen en esta pequeña web.



Por otro lado, la solución no es muy elegante, necesitamos que cada página incluya el menú de la izquierda para la correcta navegación, lo que implica que el mantenimiento de la web resulte muy costoso, pues hay que reimplementar cualquier cambio en todas las webs que trabajen con el menú cada vez que éste sufre alguna modificación de diseño y/o contenido.

5.5. Maquetación con marcos (`<frame>`)

La solución con marcos (**`<frame>`**) resulta mucho más elegante que la anterior, además de tener la gran ventaja de que no hay que duplicar los menús en cada página.

Para conseguir este efecto con los **`<frame>`** necesitamos, al menos, tres archivos html distintos:

1) uno principal, que será el que se cargue primero en el navegador y genere el espacio de trabajo distribuyendo el espacio de los **`<frames>`**, y
2) otros (dos o más) que serán los que se colocarán en los marcos generados por el primero.

Veamos el **HTML** de este archivo principal.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xHTML1/DTD/xHTML1-transitional.dtd">
<HTML xmlns="http://www.w3.org/1999/xHTML">
<head>
<meta http-equiv="Content-Type" content="text/HTML; charset=utf-8" />
<title>Maquetación con marcos</title>
</head>
<frameset cols="20%,60%,20%">
<frame frameborder="1" name="marcoIzquierdo" src="code15i.htm"/>
```

```
<frame frameborder="1" name="marcoDerecho" src="code15d.htm"/>
<frameset rows="50%,50%">
<frame frameborder="1" name="marcoDArriba" src="code15up.htm"/>
<frame frameborder="1" name="marcoDAbajo" src="code15down.htm"/>
</frameset>
</frameset>
<noframes>Información para los navegadores que no llevan frames.</noframes>
</HTML>
```

En primer lugar, es importante recordar que la etiqueta **<frameset>** es una alternativa a **<body>**; por lo tanto, éste es uno de los primeros puntos que hay que tener en cuenta cuando se trabaja con marcos. En este documento inicial no debe aparecer la etiqueta **<body>** y **<frameset>** ocupará su lugar.

El **<frameset>** define la cantidad de marcos y la distribución que éstos tienen utilizando el atributo ***cols*** (marcos horizontales) y ***rows*** (marcos verticales).

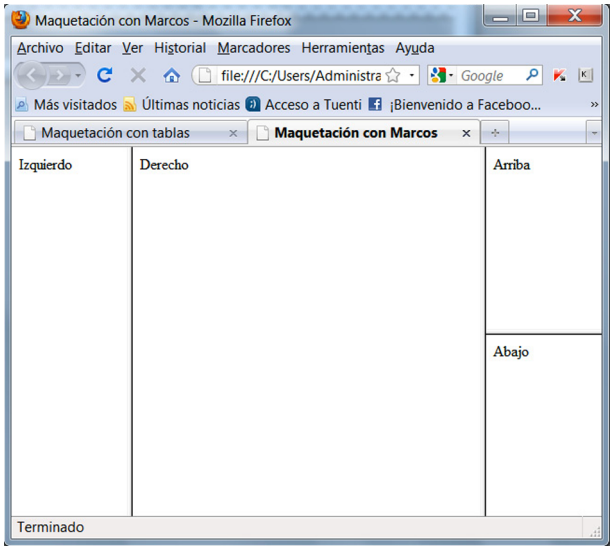
```
<frameset cols="20%,60%,20%">
```

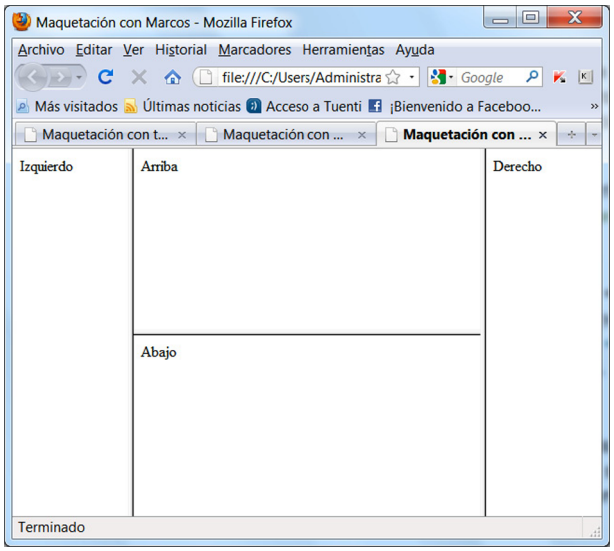
Aquí hemos dividido la página en tres columnas (**cols** ="**20%, 60%, 20%**"), la tercera la hemos subdividido a su vez en la parte superior y la parte inferior (***rows*** ="**50%, 50%**"). Para esto hemos tenido que duplicar **<frameset>**, justo en el lugar en el que aparecería el **<frame>** de la tercera columna.

```
<frameset rows="50%,50%">
```

De este modo, se indica que el tercer **<frame>** es divisible, al igual que el espacio de la página completa lo fue anteriormente.

Si colocáramos el segundo **<frameset>** una línea más arriba, entre los dos **<frame>** que le preceden, la columna que resultaría dividida sería la segunda y no la tercera como ocurre en el conjunto resultante. Si lo hiciéramos quedaría de este modo.





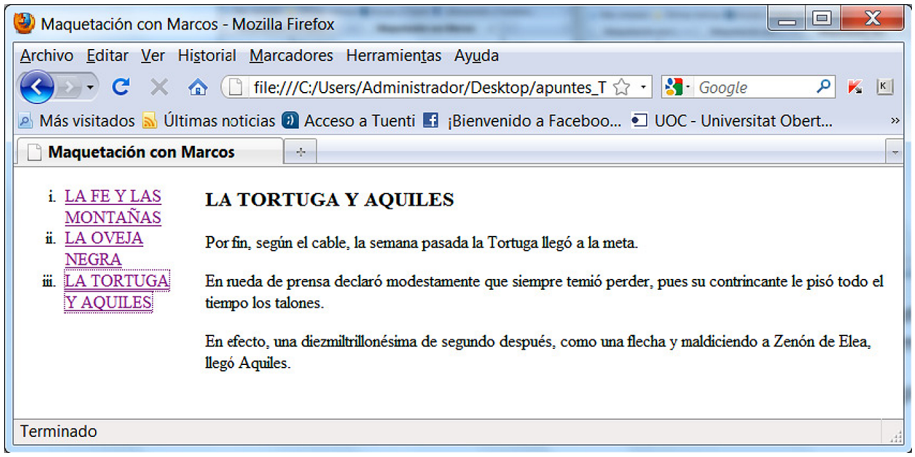
Hay que tener bien claro que una cosa es el marco o **`<frame>`** y otra la página que responde a un documento **HTM** o **HTML**, que se incluye en el marco.

`<frame>` será el marco definido y el atributo **src** indicará qué página se ha de cargar en el marco ***name***.

```
<frame frameborder="1" name="marcoIzquierdo" src="code15i.htm"/>
```

Nuestro **`<frameset>`** distribuye el espacio y luego cada **`<frame>`** particular tiene un atributo nombre (**name**) que ayuda a identificarlo y carga en principio una página web (archivo HTML) en su espacio de marco identificado por **name**.

Esta página podrá ser cambiada dinámicamente durante la interacción con la página de **`<frames>`**, y esto es lo que nos permitirá mejorar la usabilidad de la página y hacer que funcione como vimos anteriormente con las tablas **`<table>`**. Tal como podemos comprobar en el siguiente ejemplo:



Una vez disponemos de nuestro documento completo cargado en el navegador, para conseguir que los archivos de la derecha se vayan cargando, tal como vamos seleccionando los enlaces de la izquierda hemos jugado con el atributo **target** de la etiqueta **`<a>`**, que hemos colocado en el archivo de la izquierda y cuyo código podemos ver aquí:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xHTML1/DTD/xHTML1-transitional.dtd">
<HTML xmlns="http://www.w3.org/1999/xHTML">
<head>
<meta http-equiv="Content-Type" content="text/HTML; charset=utf-8" />
<title>Documento sin título</title>
</head>
<body>
<ol type="i">
  <li><p><a href="code17d1.htm" target="marcoDerecho">LA FE Y LAS MONTAÑAS</a></p></li>
  <li><p><a href="code17d2.htm" target="marcoDerecho">LA OVEJA NEGRA</a></p></li>
  <li><p><a href="code17d3.htm" target="marcoDerecho">LA TORTUGA Y AQUILES</a></p></li>
</ol>
</body>
</HTML>
```

Con el fin de indicarle al ***target*** dónde debe ubicar el archivo que se ha de cargar, utilizamos el nombre (***name***) del **`<frame>`** al que queremos enviar la página html.

El estudiante deberá experimentar y aumentar sus conocimientos a partir de otros manuales o fuentes de documentación. Aquí no disponemos de más espacio para comentar los **<frame>**, pues la idea es establecer unas bases mínimas para la comprensión de la maquetación con ** **, sus peculiaridades, ventajas y desventajas sobre otros sistemas.

Tablas, marcos y divisiones

Nada impide combinar las tres modalidades –tablas, marcos y divisiones– a gusto del autor, partir de una *site* con marcos y después utilizar tablas y divisiones en las páginas que serán cargadas. Se trata de un mundo amplio y complejo que el estudiante deberá explotar según sus intereses y circunstancias.

Ahora pasaremos a ver un modelo basado en divisiones . Si con las tablas y los marcos las hojas de estilo en cascada (HEC/CSS) no intervienen, salvo que deseemos formatear la tablas con un estilo personalizado, pasaremos a crear todo a partir de hojas de estilo en cascada (HEC/CSS) y aprenderemos qué es una división, cómo se le puede asignar estilo y cómo realizar su posicionamiento en pantalla y en la página HTML.

5.6. Preliminares a la maquetación con divisiones

5.6.1. Introducción

Una vez situados dentro de la problemática de la maquetación y navegación, y visto el modo en que esto se puede hacer con **<tables>** o con **<frame>**, ya podemos comprender el impacto que han tenido las implementaciones de CSS para la maquetación de documentos.

Ahora, centrémonos mejor para ir al grano en la etiqueta ** **, que, junto con **** y las propiedades que aportan las CSS, han dado un giro importante en la maquetación de los últimos tiempos.

**** tiene un uso más restringido que ** **, pues es un elemento que se refiere a una única línea o a un subconjunto de una línea.

** ** hace referencia a una caja multilínea, que puede contener cualquier cosa, líneas o dibujos o, a su vez, nuevas ** ** en su interior y, evidentemente, cualquier etiqueta que deseemos.

5.6.2. Definiciones de span y div

En las definición del W3C podemos ver que la única diferencia en las características de uno y otro,

```
<!ELEMENT DIV - - (%flow;)*   -- generic language/style container -->
<!ATTLIST DIV
%attrs;               -- %coreattrs, %i18n, %events -- >
<!ELEMENT SPAN - - (%inline;)* -- generic language/style container -->
<!ATTLIST SPAN
%attrs;               -- %coreattrs, %i18n, %events -- >
```

tal como señalamos, es que uno, ** **, se define como %flow y el otro es %inline.

Para entendernos, esto significará lo que comentamos anteriormente, que uno, ****, actúa en una única línea, mientras que el otro, ** **, trabaja como un "flujo" (izquierda->derecha->arriba->abajo) mediante varias líneas.

Nos centraremos aquí y ahora en la potencialidad del elemento ** **. Éste, como cualquier otra etiqueta, dispone de una serie de atributos que puede admitir, a su vez, una amplia gama de valores.

5.6.3. Herencia de atributos

Un concepto importante que hay que tener en cuenta cuando utilicemos el elemento ** ** es el concepto de "herencia". La herencia hace referencia a los atributos de una etiqueta.

Las etiquetas son contenedores, tal como comentamos al principio. Cada contenedor presupone que hay algo en su interior, aunque pueda ser un único cuadro de color blanco de un pixel. Podemos considerar que todas las etiquetas que tienen apertura y cierre van a comportarse como un contenedor.

No existe ningún problema para que haya un contenedor dentro de otro y que a su vez contenga otro en su interior.

Así, los atributos se "heredan" del contenedor al contenido. Si el contenedor tiene su atributo color con el valor #f00 (rojo), el contenido tendrá como atributo el mismo color. Si tenemos un párrafo, **<p>**, con texto, dentro de un ** ** que tenga el atributo color en el valor #f00 (rojo), el **<p>**, contenido, "heredará" ese valor.

Observación

Esto es útil si queremos utilizarlo ya desde las etiquetas <html> y <body>, de manera que sus atributos se hereden por todo el documento.

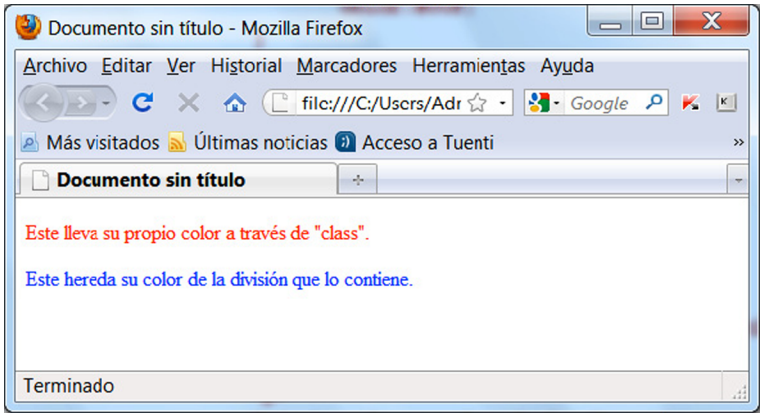
Imaginemos el siguiente código:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Herencia de atributos</title>
<style>
div.caja{
        color:#00F;
}
p.texto{
        color:#F00;
}
</style>
```

```
</head>
<body>
<div class="caja">
<p class="texto">
Éste lleva su propio color mediante "class".
</p>
<p>
Éste hereda su color de la división que lo contiene.
</p>
</div>

</body>
</html>
```

La etiqueta ** ** contiene dos párrafos, uno de ellos indica su propio color, pues tiene el atributo class especificado; por su lado, el otro no dispone de tal atributo y, en consecuencia, "hereda" el de la etiqueta ** ** en el que está contenido. Podemos observar el resultado en un documento HTML.



Entender el concepto de herencia es muy importante por dos aspectos ventajosos:

1) Primero, el de sacarle partido ahorrando reescribir atributos, pues no es necesario repetirlos si disponemos de una buena planificación, lo que nos dirá qué atributos se adjudicarán a lo largo de un documento o de una división. De ese modo no habrá que indicarlos al por menor.

2) El segundo consiste en comprender la herencia para evitar fallos en nuestros documentos, que no serán fallos ubicados en la etiqueta que estemos escribiendo, sino por el hecho de que la etiqueta en cuestión está heredando algún atributo extraño de un contenedor de la que es contenido, aunque no se perciba de manera inmediata.

Por último, hay que señalar un detalle más sobre el asunto de la herencia, y es que las hojas de estilo en cascada están pensadas para verse afectadas por el estilo que tienen más próximo al contenido que representan, como hemos visto en la última captura; el heredero (párrafo rojo) puede prescindir de su herencia al definir él mismo tal atributo.

Otro aspecto que cabe destacar de ** ** es que está concebida para una maquetación dinámica, de flujo, al contrario que las tablas y los marcos, que están más pensadas en términos de texto impreso y que son estáticas y poco redimensionables.

5.7. Manos a la obra

Una vez entendido lo anterior, comencemos a maquetar el documento a partir del elemento .

Imaginemos que deseamos un documento que disponga de una cabecera, un cuerpo y un pie de página. Mediante ** ** podemos "conceptualizarlos", de modo que la estructura del documento se mantenga independientemente del contenido que vayamos introduciendo en ella.

Cada cabecera llevará el nombre de una sección distinta según la ocasión, el cuerpo llevara el texto referido a este artículo en ese caso y el pie, datos de fecha y autoría, por poner un ejemplo específico del contenido puntual que tienen los otros dos elementos.

Algo como lo que podemos ver aquí se lograría con el siguiente código:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Maquetación con DIV 1</title>
</head>
<body>
<div class="myDOC">
   <div class="cabecera">CABECERA</div>
   <div class="cuerpo">CUERPO<br/>CUERPO<br/>CUERPO<br/></div>
 <div class="pie">PIE</div>
</div>
</body>
</html>
```

Lo primero que hay que decir es que en muchos casos el verdadero trabajo del archivo se realizará en la hoja CSS, y quizá en archivos **JS de JavaScript**, de modo que en el archivo HTML se perfilará la estructura y la dependencia jerárquica de los elementos, pero el contenido, el colorido y el diseño, así como la estructura, descansarán en otros archivos.

Observad que hemos incluido nuestros tres apartados , cabecera, cuerpo y pie, en un apartado **`<div>`** más general, denominado "myDOC", que los abarca a todos, pues es una manera de trabajar elementos comunes a los tres y de agruparlos a la hora del manejo.

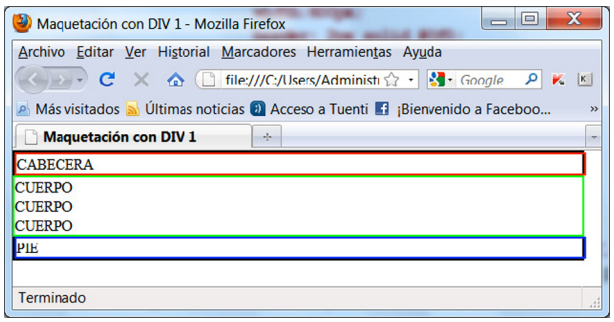
En el documento no se ve nada más que el nombre de cada DIV o espacio reservado, pues las divisiones no tienen aún contenido y además carecen de CSS que les aporte diseño y posicionamiento. Añadiremos al documento, dentro de la etiqueta <head>, la línea siguiente

```
<link href="maquetar-con-div.CSS" rel="stylesheet" type="text/CSS" />
```

para referenciar una hoja de estilo que nos ayudará en nuestro proceso de maquetación. El contenido de este archivo **maquetar-con-div.CSS** será el siguiente:

```
@charset "utf-8";
/* CSS Document */
html, body{
        margin:0px;
        padding:0px;
}
div.myDOC{
        width:600px;
        border: 2px solid #000;
}
div.cabecera{
        width:600px;
        border: 2px solid #f00;
}
div.cuerpo{
        width:600px;
        border: 2px solid #0f0;
    margin-left: -2px;
}
div.pie{
        width:600px;
        border: 2px solid #00f;
}
```

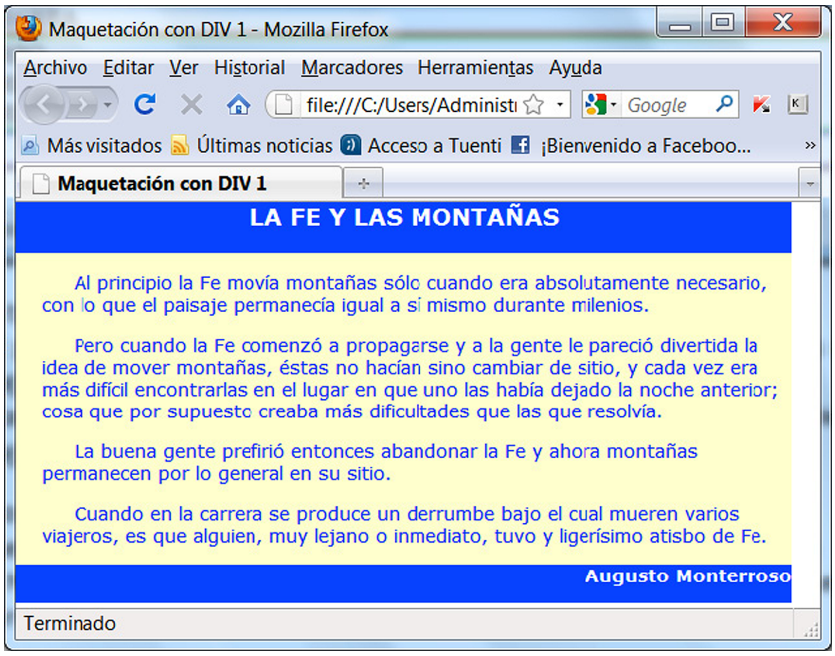
Para ver el archivo resultante podemos hacer clic aquí. Vemos un cuadro negro grande que es myDOC y dentro de él las tres partes del documento que responden a los tres apartados comentados.



Todos tienen el mismo ancho, pero observad que la cabecera y el pie exceden en 2 píxeles el final del cuerpo general "myDOC" (el que los envuelve a todos), mientras que el cuerpo se ajusta a él, pues le hemos especificado el atributo (-2px). Estos pequeños detalles son importantes porque pueden ser fuente de muchos problemas a la hora de ajustar los bloques **`<div>`**.

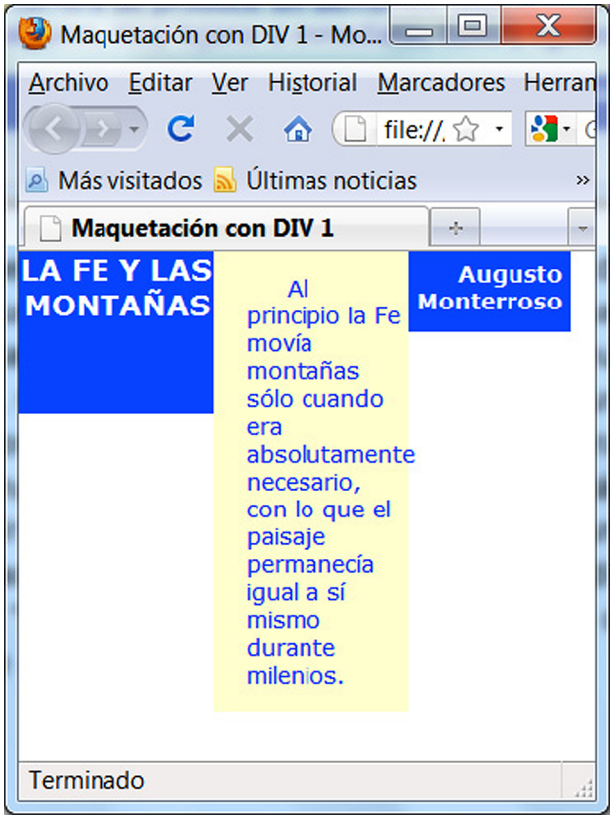
Pero, como podemos observar en este ejemplo básico, los cuadros se limitan a la altura de los elementos que contienen y el resultado es en apariencia muy poco atractivo visualmente. Mejoraremos nuestra maquetación aplicándola a los ejemplos que vimos anteriormente con **`<table>`** y **`<frame>`**.

Veamos un ejemplo más claro.



En el ejemplo ya se observa cómo hemos creado tres , cabecera, cuerpo y pie, que forman una unidad, dado que hemos situado a los tres en un **`<div>`** general que los mantiene como un bloque unitario.

Si hubiéramos hecho esto con marcos o tablas, nos encontraríamos con una estructura fija que no nos permitiría muchas variaciones o alteraciones; sin embargo, ahora podemos elegir cómo se van a distribuir nuestros **`<div>`** (incluso de manera dinámica) e inducirlos fácilmente, a partir del ejemplo siguiente:



¿Cómo el documento ha cambiado totalmente en su visualización? Sencillamente añadiendo a las distintas ** ** el atributo ***float*** con el valor "***left***", y sin tocar en absoluto nuestro código HTML, que sigue siendo exactamente el mismo.

```
div.cabecera{
        width:120px;
        text-align:center;
        color: #fff;
        font-family:Verdana, Geneva, sans-serif;
        font-size: 18px;
        background-color:#03F;
        font-weight:bold;
        height:100px;
  float:left;
}

div.cuerpo{
        width:100px;
        font-family:Verdana, Geneva, sans-serif;
        font-size: 14px;
        background-color: #FFC;
        color: #00f;
        text-indent: 25px;
        padding-left: 20px;
        float:left;
```

```
}

div.pie{
        width:90px;
        text-align: right;
        color: #fff;
        font-family:Verdana, Geneva, sans-serif;
        font-size: 14px;
        background-color:#03F;
        height:40px;
        font-weight:bold;
        float:left;
        padding: 5px;
}
```

Sencillamente hemos cambiado el apilado horizontal de las distintas por uno que le permite estirarse hacia la derecha tanto como le sea posible.

Nota

Sólo se extenderá hacia la derecha si la ventana del navegador se lo permite; de no ser así, se apilará como lo hacía anteriormente.

Trataremos de explicar con más claridad si el tipo de cliente que accede a la web es distinto (móvil, pad) o si al trabajar en una pantalla de pequeñas dimensiones la web mantendrá su legibilidad sin necesidad de andar jugando con el *scroll*:

Dado que los ** ** permiten un manejo mucho más dúctil que el que tienen marcos y tablas, el trabajo con ** ** es realmente mucho más complejo, pero aquí hemos querido poner de manifiesto el porqué de su interés en la maquetación frente a otras posibilidades.

Además, debemos señalar que no hemos tenido que hacer ninguna variación en el código HTML, que sólo hemos retocado las CSS u hojas de estilo en cascada (HEC), que indican a las distintas ** ** tanto su estilo visual como su tipo de posicionamiento mediante el atributo ***class***.

Diseño web

Hay que entender que el diseño web se enfrenta a un problema trascendente que es muy importante tener en cuenta una vez pasamos del interés aficionado al profesional. Una web comercial u organizativa necesita ser vista de muchos modos distintos y eso se ha convertido a día de hoy en un componente muy importante del diseño que la solución a partir de ** ** ha conseguido mejorar mucho.

Por un lado, el maquetado dinámico es una técnica que se ha de conocer bien y, por otro, se deben realizar comprobaciones continuas en los periféricos en los que queremos trabajar. Por las razones anteriores hemos dedicado nuestro tiempo a este ejemplo. Si obviamos lo comentado, maquetar con ** ** puede ser tan fácil como dibujar la estructura a mano en un papel, dado que siempre será posible acogerse a un maquetado estático semejante al que haríamos si lo realizáramos físicamente.

Existe un atributo de las etiquetas, ***position***, especialmente útil en ** **, que se denomina "***position***". Si asignamos su atributo al valor "***absolute***", a partir de ese momento cada ** ** que ubiquemos (también sirve para otras etiquetas, pero si no se maneja con cuidado puede dar resultados un tanto caóticos) tendrá exactamente la posición deseada con respecto a los márgenes izquierdo y superior medida en píxeles.

Con ello podríamos crear un documento (estático) que estuviera tan milimetrado como si lo hiciéramos sobre una mesa de dibujo. El documento mantendría su formato independientemente de las visualizaciones que hiciéramos de él, lo que en algunos contextos podría ser interesante para el diseñador (imaginemos una ficha o serie de fichas que tuvieran que adaptarse en la impresión a un formato de papel concreto). Animamos al estudiante a que experimente con este atributo en sus documentos, pues no disponemos de más espacio para ejemplificar el tema. Actualmente, lo más usual es contar con el flujo comentado anteriormente de las etiquetas.

Resumiendo, podemos decir que el uso de las hojas de estilo en cascada (HEC = CSS) y de la etiqueta ** ** (recordemos que **** tiene unas ventajas muy semejantes a nivel de una única línea) ha supuesto una revolución en el concepto de maquetación, frente a los ya clásicos **<frame>** y **<table>**.

Libertad de elección

Hemos podido leer multitud de diatribas sobre las ventajas y desventajas de unos (marcos, tablas y divisiones) sobre otros, pero evitaremos tomar partido; entendemos que el diseñador debe tener plena libertad de elección y no tomar el asunto como una cuestión prescriptiva, si esto supone una excesiva inversión de tiempo o coste. Elija cada uno con tranquilidad el método que mejor venga a las circunstancias con las que se verá en cada caso; lo importante, pensamos, es conocer las opciones y poder elegir.

Algo que también deberá quedar claro es que ** ** es sólo la representación de un concepto (cuadro con contenido); existen muchas otras etiquetas que pueden ser tratadas del mismo modo con el que hemos tratado ésta, siempre que dispongan de los mismos atributos (hay muchas que sí los tienen); así, las etiquetas que se definen como *inline* o de una única línea se comportarán de manera parecida a ****, mientras que las que sean *flow* lo harán como ** **.

Webs recomendadas

Discusión acerca del uso de *frames* en Internet. http://www.google.es/search?hl=es&client=firefox-a&channel=s&rls=org.mozilla%3Aes-ES%3Aofficial&hs=tNC&q=why+not+frames&btnG=Buscar&meta=&aq=f&oq=

W3C Recommendation. Cascading Style Sheets, level 2 CSS2 Specification, http://www.w3.org/TR/2008/REC-CSS2-20080411/

Lecturas recomendadas

Autores varios (2008). *Profesional CSS para diseño Web*. Madrid: Anaya Multimedia.

J. Cranford Teague (2005). *DHTML y CSS*. Madrid: Anaya Multimedia.

Ch. Schmitt (2007). *Curso CSS*. Madrid: Anaya Multimedia.

R. York (2006). *CSS práctico*. Madrid: Anaya Multimedia.

6. Elementos avanzados

6.1. Objetivos

El objetivo de este apartado es que el estudiante obtenga una visión general de los elementos avanzados que acompañan habitualmente a HTML, de su principal utilidad y características. En este sentido, que sepa dónde obtenerlos y cómo dar con ellos es el primer paso para poder dominarlos e incorporarlos a sus páginas web.

6.2. Contexto de trabajo

Hay que tener siempre en mente, si se quiere diseñar con elegancia, el ciclo de vida y el funcionamiento de una página web. Resulta muy común seleccionar un editor WYSIWYG (*what you see is what you get*) y crear una página maravillosa visualmente, pero que luego ésta tenga verdaderas dificultades para estar situada en el lugar que le corresponde (en la WWW) y funcionar de la manera esperada.

La vida de la web discurre entre nuestro ordenador (el cliente) y el ordenador en el que la página esté alojada (servidor). Aunque no hay ningún inconveniente para que este ordenador sea el mismo, lo habitual es que se trate de ordenadores diferentes.

Una vez escrita la página web, la depositamos en el servidor. Cuando alguien la quiere visitar accede a ella (al servidor) con su navegador (cliente). El servidor la manda al cliente y éste la visualiza.

En este contexto, imaginemos que nuestra página web consiste en un conversor de euros a pesetas (a algunos de los que aprendimos a hacerlo en pesetas todavía nos cuesta valorar en euros).

6.3. JavaScript

El **HTML** permite que los elementos de una página estén conceptuados o semantizados. Las hojas de estilo en cascada (**HEC**), **CSS**, los dotan de aspecto visual y de posicionamiento, pero ¿cómo conseguimos que se puedan hacer las operaciones que comentábamos de trasvasar de una moneda a otra? Para esto necesitamos los siguientes pasos:

a) Una entrada de datos por parte del usuario.
b) Hacer el cálculo de la operación.

c) Una salida de datos.

Los puntos *a* y *b* no presentan problema, pero ¿cómo realizamos el cálculo? Naturalmente este deberá hacerlo el ordenador, pero ¿qué ordenador, el cliente o el servidor, según el contexto que establecimos al principio?

Ante esto tenemos dos modelos:

a) Una vez introducidos los datos se envían al ordenador servidor, que calcula y devuelve una nueva página con el resultado (**PHP**).
b) Hacemos que nuestro ordenador (cliente) haga los cálculos (**JavaScript**).

La primera solución está mal pero implica que un servidor se ocupe de todos los cálculos de las páginas que contiene o gestiona, que pueden llegar a ser muchas. A su vez, esto supone una gran carga para el servidor, mientras que todos los ordenadores cliente que lo visitan en ese momento tendrían un gran contingente de potencia de proceso desaprovechada.

Sin embargo, este primer modelo funcionará bien si lo que nos interesa es que la página no sea un elemento estático igual para todos los usuarios, pues podemos personalizar la página en el servidor (con PHP) antes de enviarla al cliente (como HTML) y hacer que cada uno vea una página distinta en función de sus necesidades u opciones. Éste es el modelo que sigue **PHP**, **ASP** y **PERL**. Más adelante echaremos un vistazo a PHP como ejemplo de estos tres casos.

El segundo modelo es mucho más práctico si se trata únicamente de pequeñas operaciones o de cambios en la página que no requieran nuevos datos del servidor.

Imaginemos que queremos que un botón cambie cuando pasamos el cursor por encima de él. Podemos tener ya nuestras dos imágenes (antes y después del ratón) cargadas y mostrar una u otra a voluntad, sin la necesidad de realizar peticiones al servidor de la nueva imagen o cargar la página entera de nuevo.

Éste es el espacio de operatividad que ocupa **JavaScript**. Es un lenguaje de programación incorporado al código de la página HTML y ejecutado en el cliente, y puede realizar cálculos, además de modificar el **DOM**[10] del documento.

(10)El DOM sería la "imagen en un árbol nodal" del documento que tiene el navegador-cliente, lo que implica la posibilidad de actuar tanto sobre el HTML como sobre CSS, que son los elementos que podemos encontrar en el DOM para recuperar o modificar.

Como hemos visto, JavaScript es el lenguaje de programación nativo del HTML. Para hacer un estudio en intensidad de JavaScript, de sus posibilidades y capacidades, os recomendamos que reviséis la documentación de la especificación de JavaScript.

Allí podréis ver todas las posibilidades que tiene con ejemplos prácticos y todo lujo de detalles.

Aquí, sencillamente, a partir de un ejemplo práctico y básico, trataremos de explicar los rudimentos del JavaScript para que el estudiante se haga una idea general.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Cálculo</title>
<script language="javascript" type="text/javascript">
    function traduce(){
var valorP = document.ejemploJS.pelas.value;
var valorE = document.ejemploJS.eurito.value;
if(valorP > 0){
         valorE = valorP/163;
         alert( valorP + "Pesetas, son \n" + valorE + "Euros.");
}
else{
         valorP = valorE * 163;
         alert( valorE + "Euros, son \n" + valorP + "Pesetas.");

}
}

function meVoy(){
         alert("Adiós me voy a GOOGLE...");
         document.location = "http://www.google.com";
}

</script>
</head>

<body>
<script language="javascript" type="text/javascript">
document.write("<h1>El título lo ponemos directamente en JavaScript</h1>");
</script>
<h2>Este título se ha realizado con HTML</h2>
<form name="ejemploJS" method="post" action=""
onsubmit="traduce()">
<label>Pesetas</label>
<input type="text" name="pelas" value="0" /><br/>
<label>Euros</label>
<input type="text" name="eurito" value="0" /><br/><br/>
<input type="submit" value="Traduce" />
<input type="reset" value="Reset" />
</form>
```

```
<a href="javascript:meVoy();"> Quiero ir a otro sitio... </a>
</body>
</html>
```

Tras el encabezamiento que ya conocemos, en la sección <head> encontramos el *script* de JavaScript. Puede ir en cualquier lugar del documento envuelto en las etiquetas

```
<script></script>
```

Es habitual acompañar el *script* de los atributos

```
language="javascript", que indica el lenguaje de programación.
type="text/javascript", que indica el tipo de texto contenido en el script.
```

El **<script>** no se visualiza en el documento, pero forma parte de él; no hay ningún problema, tal como hemos visto anteriormente con las hojas de estilo en cascada CSS, en situarlo en un archivo externo que se cargará en el documento en tiempo de ejecución, cuando el navegador cargue la página.

Para usar un archivo externo en lugar de escribirlo en **<head>** lo referenciaremos del siguiente modo.

```
<script language="JavaScript" type="text/JavaScript" src="js/miarchivoJS.js"></script>
```

Observad que el **<script>** está vacío. En este caso, el contenido con las instrucciones estaría dentro del archivo miarchivoJS.js, en el subdirectorio /js, dependiente de la misma raíz del HTML con el que estemos trabajando.

Recomendación

A este respecto, os recomendamos que repaséis este lenguaje de programación antes de lanzaros a complicados desarrollos de funciones. Por otro lado, no resulta muy difícil buscar las funciones que necesitamos en algún documento de la web y copiarlas y pegarlas en la nuestra para luego adaptarlas a nuestros intereses, siempre que seamos capaces de leer correctamente el código para poder efectuar las adaptaciones oportunas.

Aunque no exclusivamente, lo que suelen contener los archivos JS o los **<script>** son funciones escritas siguiendo la sintaxis de JavaScript, que no es muy distinta de la sintaxis de otros lenguajes de programación, aunque tiene sus peculiaridades.

Hemos definido en nuestro archivo una función llamada traduce(); los paréntesis nos señalan que se trata de una función, mientras que las instrucciones que deberá ejecutar se encierran entre llaves {}.

```
function traduce(){
var valorP = document.ejemploJS.pelas.value;
var valorE = document.ejemploJS.eurito.value;
if(valorP > 0){
         valorE = valorP/163;
         alert( valorP + "Pesetas, son \n" + valorE + "Euros.");
}
else{
         valorP = valorE * 163;
         alert( valorE + "Euros, son \n" + valorP + "Pesetas.");

}
}
```

La ejecución de la función es sencilla para aquellos que estén familiarizados con los lenguajes de programación; si no es así, es conveniente profundizar, tal como hemos indicado, un poco en el tema. Antes de continuar daremos por sentado que se conocen los rudimentos de la programación para no extendernos en la explicación de los conceptos fundamentales de ésta.

Se declaran variables a partir de la palabra *var* y un nombre y se les asigna un valor a partir del signo igual:

```
var valorP = document.ejemploJS.pelas.value;
var valorE = document.ejemploJS.eurito.value;
```

lo peculiar es que los valores los tomamos del formulario en HTML que hemos creado en la página

```
<form name="ejemploJS" method="post" action="" onsubmit="traduce()">
<label>Pesetas</label>
<input type="text" name="pelas" value="0" /><br/>
<label>Euros</label>
<input type="text" name="eurito" value="0" /><br/><br/>
<input type="submit" value="Traduce" />
<input type="reset" value="Reset" />
</form>
```

los valores de las variables los tomamos del documento actual **document**, del formulario **ejemploJS** y del nombre del input **pelas o eurito**, y así componemos la variable **document.ejemploJS.pelas.value;** que ha sido cargada cuando el usuario hace clic en "Traduce" a través de la operatividad del formulario **`<form>`**.

Esta variable **document.ejemploJS.pelas.value;** se ubica en lo que se conoce como el Document Object Model (DOM), que sería la imagen nodal del documento que guarda el navegador, para poder generarlo y trabajar con él.

Web recomendada

Para más información sobre DOM, podéis consultar:
http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-1590626202.

Trabajamos las variables con un **if(valorP > 0)** y algunas operaciones numéricas con las variables y acudimos a una función muy socorrida en **JavaScript** "**alert**()" que nos permite mostrar el resultado en una ventana sin más complicaciones.

Aunque éste es el modelo más sencillo, se puede hacer mucho más que todo esto, se puede dinamizar la página a voluntad, siempre que nos mantengamos en los elementos disponibles en el documento existente en nuestro cliente o navegador.

Un ejemplo muy interesante y que enlaza con lo que aprendimos antes sobre ** ** sería generar un documento con ** ** vacías, que iremos llenando dinámicamente con funciones JavaScript, disparadas desde enlaces o formularios que irán cambiando el contenido interno de los ** ** establecidos y mostrando distintos documentos o documentos modificados acorde a las interacciones del usuario.

JavaScript server-side

Existe ya una tecnología JavaScript server-side que sería la correspondiente a la JavaScript habitual del cliente, pero no le dedicaremos aquí espacio, aunque siguiendo el enlace el estudiante podrá ampliar sobre el tema.

El uso de **<form>** y su funcionamiento puede resultar en principio un tanto complicado para quien comienza en este campo, así que hemos añadido una función mucho más sencilla para poder probar el funcionamiento del esquema de trabajo de JavaScript:

```
function meVoy(){
         alert("Adiós me voy a GOOGLE...");
         document.location = "http://www.google.com";
}
```

Notad lo sencillo que resulta desencadenarla tan sólo desde un simple enlace:

```
<a href="javascript:menVoy();"> Quiero ir a otro sitio... </a>
```

no hay más que crear la función en un *script* y asignarla como valor en el atributo **href** de la etiqueta **<a>**.

Por último, hemos añadido también un ejemplo de cómo se puede escribir el propio HTML desde un **<script>** JavaScript incluido entre el HTML del documento para que se vea otra de las capacidades más elementales:

```
<script language="javascript" type="text/javascript">
document.write("<h1>El título lo ponemos directamente en JavaScript</h1>");
</script>
```

En principio, resulta evidente que es mucho más sencillo escribir directamente el **HTML**; sin embargo, podría interesarnos escribir distintas cosas con document.write(), según qué eventos o situaciones, evaluando una expresión. En este caso, hacerlo así ya no resulta estático, sino dinámico y mucho más creativo y abierto, como podemos inducir en el siguiente código basado en una decisión:

```
<script language="javascript" type="text/javascript">
if(expresion){
          document.write("<h1>Sí</h1>");
          }
else { document.write("<h1>No</h1>"); }
</script>
```

Haremos un pequeño repaso de la situación para el estudiante que decida ponerse manos a la obra:

1) Si necesitamos operar con valores de manera dinámica y no podemos insertarlos previamente en el HTML estático de nuestra página, resulta recomendable utilizar JavaScript.

2) La etiqueta que nos permitirá hacerlo es **`<script>`**.

3) Disponemos de tres modalidades:

a) el **`<script>`** en **`<head>`**;
b) incluirlo en **`<body>`**; o
c) hacerlo en un archivo externo.

4) Los modos más habituales de activar la funciones son el uso de formularios y de enlaces href.

5) La variables que declaremos tienen el ámbito del documento HTML (incluidos los referenciados en <head>).

6) Para que documentos HTML distintos compartan estas variables hay que utilizar las URL enviadas o las variables de formulario, así como una metodología específica para capturarlas en el documento de llegada. Se recomienda profundizar en el tema antes de lanzarse.

7) La función alert() es una buena manera de depurar código, pues nos permite averiguar el valor de una variable en un momento determinado.

Somos conscientes de que las instrucciones son escasas y es necesaria más documentación para obtener una idea clara de todo lo que se puede realizar con JavaScript, pero el alcance de este documento es limitado y resulta difícil sintetizar tan gran cantidad de información. No obstante, los enlaces que hemos ofrecido ayudarán mucho si el estudiante decide lanzarse a la programación de elementos JavaScript; existen muchos manuales y ejemplos que nos ayudarán.

6.4. DHTML

DHTML es el acrónimo de **Dinamic Hypertext Markup Language**. En realidad, no es una tecnología nueva o diferente. En su momento aparecieron las primeras versiones de JavaScript y CSS, lo que hizo que el estático HTML diera un paso adelante y comenzara a usar estas tres tecnologías a un tiempo: **HTML +JavaScript+CSS**.

Frente al documento estático que se podía encontrar por aquel momento en la WWW, esta combinación de tecnologías reconvirtió el estatismo del HTML en algo dinámico, de ahí que se le otorgara ese nombre *HTML Dinámico*.

Como podremos ver, DHTML no es una cosa distinta de JavaScript; en realidad recibió ese nombre la aplicación intensiva de JavaScript+CSS para la navegación, el diseño y el manejo de documentos HTML.

En los mismos años aparecieron el lenguaje **VBScript** y los **ActiveX**, tecnología muy semejantes a JavaScript, pero que hemos obviado en este documento porque son tecnologías propietarias de una empresa y que requieren la subordinación a los intereses de ésta para su utilización, al contrario del resto de tecnologías recomendadas en esta asignatura, que son de uso abierto y gratuito.

6.5. Java

Lo primero que hay que decir es que **Java** no tiene absolutamente nada que ver con JavaScript, por mucho que el nombre parezca indicar lo contrario.

La tecnología Java trabaja tanto en la posición *a*), con *servlets* que se ejecutan en el servidor y se comunican con la página cliente, como en la posición *b*), con *applets* que están insertos en las páginas HTML del cliente pero con capacidades múltiples, tantas como un programa de ordenador realizado en C o C ++; de hecho, en muchas ocasiones algunos de los programas no-web que utilizamos están programados con Java y permiten trabajar sobre una **Máquina Virtual Java** en nuestro ordenador.

Java es incluso capaz de crear utilidades que ponen en relación *applets* del cliente con *servlets* del servidor que mantienen comunicación y actuaciones en ambas partes de la dinámica de la web.

Para programar con JAVA es necesario un **SDK** (*software development kit*) y resulta un lenguaje compilado y no de-texto, tal y como hemos visto hasta ahora.

El SDK permite crear las fuentes de las aplicaciones y éstas se compilan en **APPLETS** o **SERVLETS** que se insertan en las páginas web como objetos independientes (<object>).

Enlace principal de la tecnología JAVA

El enlace principal de la tecnología JAVA es la web propietaria http://www.sun.com y en ella podemos encontrar muchos elementos que descargar si estamos interesados en avanzar en este tipo de programación. Elementos JAVA.

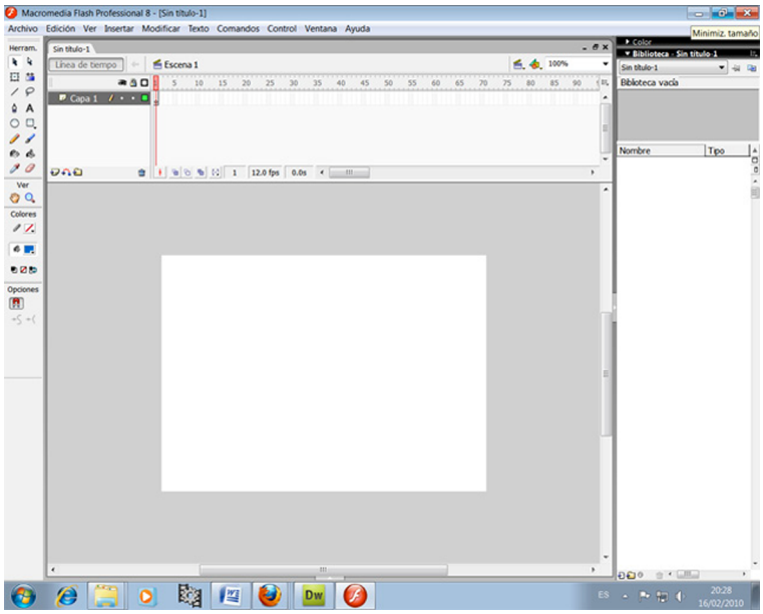
6.6. Flash

Flash, a diferencia de todo lo anterior, no es una utilidad gratuita y está vinculada a que dispongamos del programa **Flash Macromedia** en alguna de sus versiones. Es muy común en las páginas web y por ello no dejamos de incluirla en este apartado de recursos avanzados.

En el mejor de los casos se utiliza (a veces en exceso) para dar vistosidad a las páginas y en el peor, para realizar una página única y exclusivamente en un cuadro de Flash con muchos movimientos y navegaciones fascinantes, pero muy lejanas del objetivo que entendemos nosotros que debe orientar la WWW. Aunque, como en todos los casos, si uno cree que lo que necesita es un Flash, pues adelante.

Para poder programar con Flash debemos adquirir el programa o, de momento, descargar una versión de prueba en la siguiente dirección: http://www.adobe.com/downloads/.

Su interfaz es complicada, pero con algunas prácticas se pueden conseguir resultados interesantes en unas cuantas sesiones.



6.7. PHP

PHP es la tecnología de SCRIPT que se ha impuesto en la red en lo tocante a aplicaciones o trabajo *server-side* (del lado del servidor). Aunque puede funcionar en servidores como Apache en formato CGI-BIN o como módulos del servidor, lo más habitual es que se utilice en formato SCRIPT y sólo texto por decirlo de una manera sencilla.

Las siglas PHP significan *hypertext pre-processor*, curiosamente, pues su significado original estuvo más acorde con las siglas (*personal home page*). Sin embargo, el primer nombre hace más comprensible su funcionamiento. El PHP es un **lenguaje interpretado o preprocesado por el servidor**, al contrario que anteriormente con el JavaScript, que era un lenguaje interpretado por el cliente. Se recomienda al estudiante revisar la diferencia entre lenguajes de programación compilados e interpretados. Tanto PHP como JavaScript funcionan

interpretados y en modo texto. No hay que crear un archivo especial (binario, compilado) para que funcionen. Sus archivos son de texto plano y se pueden crear con cualquier editor básico de texto.

Los archivos que contienen código PHP alojados en el servidor llevan la extensión .php. Y en su interior normalmente puede haber tanto código HTML normalizado como elementos PHP.

Un documento PHP que sólo incluya código php comenzará por

```
<?php
```

y finalizará su código por

```
?>
```

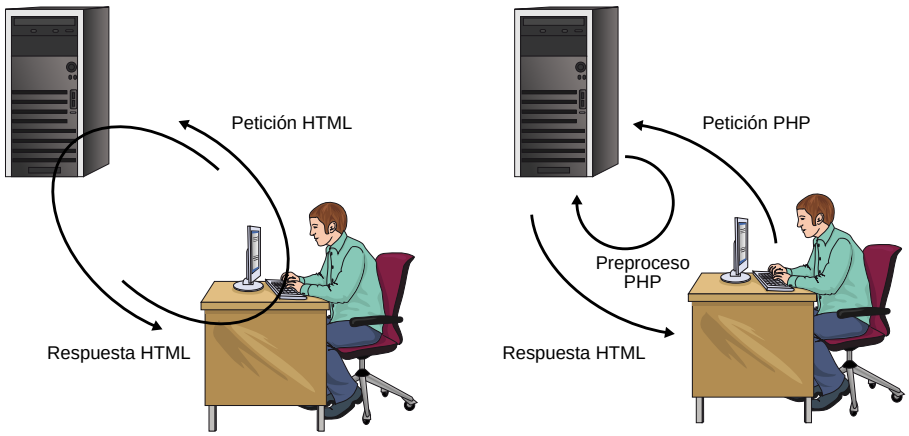
Veamos un ejemplo sencillo:

```
<?php
echo "Hola Mundo";
?>
```

Enlacemos aquí este documento y hagamos clic en él para comprobar que no se ve absolutamente nada si lo estamos ejecutando en nuestro ordenador.

Evidentemente, necesitamos que el archivo esté alojado en un servidor para que sea éste el que procese el archivo y la instrucción **echo** y que, como resultado, en nuestro cliente-navegador aparezcan las palabras "Hola Mundo".

¿Qué es lo que ha ocurrido? A diferencia de la hoja HTML, que ha sido enviada por el servidor al cliente y éste la ha procesado, en el caso del PHP lo que ha ocurrido es que el servidor ha procesado el archivo .php y ha enviado el resultado (.html) a nuestro cliente-navegador, que visualiza el producto del proceso realizado con PHP. Veamos un gráfico:



¿Cuál es la gran ventaja de todo esto? El código se puede generar en el instante en el que el cliente se conecte, no es necesario crear páginas que queden alojadas en el servidor, sino páginas capaces de generar código HTML (montar un archivo), y añadir en él el código, en ese momento y acorde a una serie de variables (quién es el usuario, qué cliente se usa, qué intención tiene el usuario, en qué sección de nuestra *site* nos encontramos), de manera que cada usuario está viendo una página diferente generada a partir de una serie de funciones PHP que se ejecutan en el servidor.

Pondremos un sencillo ejemplo.

Imaginemos que estamos trabajando con HTML normal, nos conectamos y ponemos en el navegador http://www.mipagina.edu/index.html

y en ese momento el servidor envía está página al cliente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Documento sin título</title>
</head>
<body>
<p>Hola mundo</p>
</body>
</html>
```

Y el cliente la visualiza.

En el caso de PHP lo que haríamos sería conectar con http://www.mipagina.edu/index.php y el index.php contendría un código semejante al siguiente:

```
<?php
echo "<!DOCTYPE html PUBLIC '-//W3C//DTD XHTML 1.0 Transitional//EN'
'http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd'>\n";
echo "<html xmlns='http://www.w3.org/1999/xhtml'>\n";
echo "<head>\n";
echo "<meta http-equiv='Content-Type' content='text/html;
charset=utf-8' />\n";
echo "<title>Documento sin título</title>\n";
echo "</head>\n";
echo "<body>\n";

echo funcion_que_devuelve_HTML($saluda);

echo "</body>\n";
echo "</html>\n";
?>
```

Como se puede apreciar, no existe directamente el código HTML, sino que las funciones de PHP, como echo (mostrar texto en pantalla) y otra hipotética que hemos inventado, **funcion_que_devuelve_HTML($saluda)**, serían las encargadas de generar el HTML en el servidor que después se enviaría al cliente (nuestro navegador) ya como un página normal, pero, por decirlo de una manera sencilla, "cocinada" previamente en PHP.

La sintaxis de las funciones de PHP (con evidentes diferencias) es muy semejante a la de JavaScript, de modo que si hemos aprendido este lenguaje, luego no habrá más que hacer la adaptaciones oportunas y muchas librerías creadas en JavaScript podrán ser fácilmente transcritas a PHP.

Para trabajar con PHP necesitamos que la máquina que lo ejecute cumpla las siguientes condiciones:

1) Servidor de archivos (normalmente Apache) instalado.

2) Módulos, CGI o **PHP** –texto activado en el servidor Apache para que se puedan ejecutar las funciones PHP.

3) Base de datos MySQL que, aunque no es estrictamente necesaria con PHP, a poco que se practique se hace imprescindible.

Para este fin podemos encontrar en la Web paquetes que son capaces de convertir nuestro ordenador personal en un servidor de prácticas, con un simple golpe de ratón, y al cargar un programa obtener todas estas prestaciones. De este modo, podemos realizar las prácticas que consideremos oportunas antes de enviarlo todo a un servidor externo situado en Internet o a un *hosting* que podemos obtener gratuito o de pago.

Las dos herramientas que cumplen esta condición en sistemas WINDOWS son:

- EasyPHP
- WAMP

Dejamos aquí los enlaces para que el estudiante pueda experimentar e introducirse en el interesantísimo mundo del PHP, una vez haya asimilado las tecnologías más básicas, como HTML, CSS y JavaScript

Webs recomendadas

Core JavaScript 1.5 Reference. https://developer.mozilla.org/en/Core_JavaScript_1.5_Reference

W3C Recommendation. *The DOM Structure Model*. http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html

Server-Side JavaScript Reference v1.2. http://research.nihonsoft.org/javascript/ServerReferenceJS12/index.htm

Java SE Downloads. http://java.sun.com/javase/downloads/index.jsp

Adobe. http://www.adobe.com/

Flash descarga. http://www.adobe.com/downloads/

EasyPHP (Apache, PHP y MySQL). http://www.easyphp.org/

Wampserver (Apache, PHP y MySQL). http://www.wampserver.com/

Lecturas recomendadas

Autores varios (2002). *Creación de sitios Web con PHP4*. Madrid: McGraw Hill.

J. Bobadilla; S. Alonso (1999). *HTML Dinámico a través de ejemplos*. Madrid: Ra Ma.

Ch. Cosentino (2004). *Esencial PHP*. Madrid: Prentice Hall.

E. D'Andrea (2005). *Curso profesional de programación PHP5*. Barcelona: Infor Book's.

J. Frentzen; H. Sobotka (1999). *Superutilidades para JavaScript*. Madrid: McGraw Hill.

T. Negrino; D. Smith (2001). *JAVASCRIPT*. Madrid: Prentice Hall.

J. Peña; C. Vidal (2001). *FLASH 5*. Madrid: McGraw Hill.

7. Evaluación de recursos digitales en línea

7.1. Introducción: la necesidad de evaluar

La facilidad de publicación que permite Internet provoca que proliferen páginas web sobre prácticamente cualquier tema. El impacto que está teniendo este sistema, especialmente en el ámbito académico, de investigación y en la empresa, ha provocado que instituciones y profesionales de la comunicación, información y documentación, o del diseño, desarrollen **técnicas y métodos de evaluación** de recursos digitales en línea como herramientas de control de calidad de la información, el diseño y el acceso a los recursos existentes en la Red.

La necesidad de la evaluación de recursos digitales en línea se puede enfocar desde varias perspectivas:

- por un lado, es aplicable a la **evaluación de recursos externos** para empresas, bibliotecas, centros de documentación y universidades que incorporen directorios y bases de datos de recursos web que satisfagan unos criterios de calidad como un servicio para sus clientes o usuarios;
- por otro lado, como el **diseño o la auditoría** de sitios web para mejorar sus características de navegación y de organización de la información, y obtener así una mejor posición en los resultados de los motores de búsqueda.

En la **literatura científica** se observa que los autores aportan varios motivos a la hora de establecer la necesidad de una evaluación de los recursos de la World Wide Web:

Laura M. Boyer destaca la rápida proliferación de fuentes y la falta de control real en algunos sitios, sin que intervenga ningún revisor, editor, etc.

Web recomendada

L. M. Boyer (1998). "Evaluation of Web resources" [en línea], mayo, rev. 2000, feb. 15. http://wwwlibrary.csustan.edu/lboyer/webeval/webeval.htm.

Codina asegura que es necesaria la evaluación porque, según se extiende la publicación a través de la World Wide Web, son necesarios profesionales que separen el ruido de la verdadera información, y éstos necesitan a su vez unos criterios claros y funcionales para realizar esas labores.

Lectura recomendada

Ll. Codina Bonilla (2000). "Evaluación de recursos digitales en línea: conceptos, indicadores y métodos", *Revista Española de Documentación Científica* (núm. 1, vol. 23, pág. 9-44).

7.2. Objetivo

El objetivo principal de este capítulo es la evaluación de recursos de información en línea. Se estructurará en un apartado conceptual y en la descripción de parámetros e indicadores para proporcionar los conocimientos y protocolos de actuación necesarios de identificación que ayuden a la evaluación de recursos digitales y fuentes de información de calidad publicadas en Internet.

7.3. Definición

Evaluar es determinar el valor de algo. La **evaluación**, por lo tanto, es el proceso que permite decidir sobre el valor de una cosa.

En este contexto, por evaluar entenderemos el procedimiento mediante el cual se intenta determinar, de la manera más amplia posible, el valor o la calidad de un sitio web o un recurso digital en su conjunto.

7.4. Metodologías de evaluación

Según Codina, un **método** de evaluación de recursos digitales en línea debe ser útil, racional y compatible con nuestros conocimientos en materia de sistemas de información y sistemas de información documentales.

Janet Alexander y Martha Tate sugieren un método de evaluación que incluye los siguientes pasos:

- Paso 1. Identificar el tipo de página web.
- Paso 2. Usar una lista apropiada de criterios
- Paso 3. Basándose en esta lista, determinar la calidad relativa de la página web.

Concluyen afirmando que las técnicas de evaluación están todavía en desarrollo, que la tecnología no ayuda a la creación de normas y directrices y que el establecimiento de un proceso debe ser evolutivo.

El método que propone Codina es bastante similar. Tras fijar una lista de parámetros de evaluación, bajo un criterio racional y justificado, se establecen unos procedimientos para medirlos o unas escalas de calidad. Después se deben determinar criterios de exclusión y asignar prioridades a los parámetros. Por último, hay que indicar procedimientos para otorgar puntuaciones a cada parámetro.

Web recomendada

J. E. Alexander; M. A. Tate (1996*c*). *The Web as a Research Tool: Evaluation Techniques* [en línea], 1996-1998. http://www.media-awareness.ca/english/resources/educational/teaching_backgrounders/internet/web_as_research_tool.cfm

Sobre los criterios de exclusión considera que si un recurso no alcanza a cumplir con ciertos parámetros, no es susceptible de ser evaluado. Las prioridades y los niveles se establecen según el escenario que se debe evaluar; bien sea para la creación de directorios, bien para agencias de evaluación o auditorías internas o externas, se puede dar más peso a unos indicadores o a otros.

Por último, se deben construir unas **tablas** en las que se asignen puntos a cada parámetro para luego sumarlos y obtener una puntuación global para una página web. El establecimiento nominal de escalas de calidad no es esencial al método, pero es necesario ponderar los indicadores para establecer una valoración.

7.5. Criterios de evaluación

Janet Alexander y Martha Tate aseguran que algunos criterios de evaluación del material impreso son aún apropiados para la evaluación de recursos web, pero que necesitan una revisión y una adaptación.

Estos cinco criterios tradicionales en el material impreso son:

1) Precisión o fiabilidad de la fuente: hace referencia a la cualidad de la información de ser fidedigna y no contener errores. También se debe tener en cuenta si existe un editor o alguien que se haya encargado de revisar la obra.
2) Autoridad: se considera la cualificación del autor para escribir sobre el tema, así como la buena consideración de la persona que publica la obra.
3) Objetividad en el planteamiento: se debe observar si la información presenta algún sesgo, o en qué grado la información intenta inclinar la opinión de la audiencia.
4) Actualidad: es importante si el contenido del trabajo está puesto al día y si la fecha de publicación está claramente indicada.
5) Cobertura: interesa saber el número de temas incluidos en el trabajo y si éstos están tratados con exhaustividad.

Sin embargo, también es necesario considerar otros factores esenciales de las páginas web:

- El **contexto**, que pretende situar el recurso que hay que evaluar en un contexto caracterizado por la adecuación a la **audiencia** a la que va dirigido y que cuenta con factores como el alcance o el modo de tratar un tema, la audiencia y la autoría.
- El **contenido**, como punto principal en la evaluación de los recursos, desglosado en criterios que evalúan el rigor en el tratamiento de la información, el volumen, su actualización, la objetividad con la que se expone y la originalidad que éste tiene en cuanto a que aporta un conocimiento novedoso.
- El **acceso a la información** y la **interactividad** constituyen un aspecto totalmente novedoso, ya que son cualidades de Internet de las que el mundo

impreso carece. Se deben analizar la descarga de información y sus condiciones de acceso, la accesibilidad y capacidad de uso, las restricciones que se puedan imponer en el acceso, la presencia o ausencia de una política de cobro por el acceso a la información, los requisitos especiales relacionados con el hardware y el software, la problemática en torno a los derechos de autor, la capacidad de recuperación de la información y su capacidad de uso, la interactividad y la calidad de los enlaces del hipertexto.

- La **popularidad o el impacto**, también denominado **visibilidad** o número de páginas web que mantienen enlaces hacia la web objeto de estudio, ha sido equiparada con las citas bibliográficas recibidas y ha centrado la atención de una corriente de estudio que pretende aplicar las técnicas de la bibliometría[11] a estos nuevos documentos.

(11)La bibliometría aplica métodos matemáticos y estadísticos a la producción de literatura científica, con el objetivo de estudiar y analizar los resultados de la actividad científica. Podéis consultar CybermetricsLab, CCHS-CSIC. Disponible en: http://internetlab.cindoc.csic.es/index.asp

- La **autodescripción** hace referencia a la descripción que se realiza en una página de ella misma. Una buena descripción favorecerá la recuperación de la página por medio de los motores de búsqueda y las agencias de evaluación, fundamentalmente mediante el título y las metaetiquetas y los metadatos.

- El **diseño** y la **estética**, que se traducen en **legibilidad** y **ergonomía** y aportan la estructura del documento.

- Los **servicios** que se ofrecen y que caracterizan a este medio interactivo.

7.5.1. El contexto

El primer grupo de criterios pretende situar el recurso que se ha de evaluar en un **contexto**. Es necesario saber cuáles son los **objetivos** de la página web que vamos a evaluar, así como la **audiencia** a la que aquélla va dirigida, y establecer si puede ser útil para nuestros usuarios. Estos datos pueden ser definitivos para continuar o no con el proceso de evaluación.

Sin embargo, establecer el contexto de un recurso de la web puede ser difícil. Incluimos varios elementos que hay que tener en cuenta en este sentido: el **alcance**, la **audiencia** y la **autoridad**.

- Entendemos por **alcance** tanto el modo de tratar el tema como los objetivos que se persiguen con la publicación de esa página web. Respecto a los objetivos, hay que identificar cuáles son, ver si están claramente establecidos en la propia página y comprobar si el recurso cumple los objetivos que se propone.

- La **audiencia**/los **usuarios**. Partiendo de la necesidad de conocer a nuestros usuarios y sus necesidades de información, si consideramos una página web como un servicio que les ofrecemos, debe tener en cuenta a quién

va dirigido. La audiencia está muy relacionada con los objetivos perseguidos. Debemos preguntarnos cuál es la audiencia a la que el recurso va dirigido, si el nivel es adecuado para ésta y si el recurso satisfará las necesidades de los usuarios previstos.

- La **autoría**: **responsabilidad** y **solvencia**. En general, se refiere a si el material es una creación de una persona u organización a la que se le reconoce conocimiento definitivo en un área temática concreta. Uno de los factores que han contribuido a extender la popularidad de la Web es la facilidad con la que casi todo el mundo puede publicar en ella. Este factor, aunque es uno de sus puntos fuertes, supone un reto en la evaluación. Es difícil verificar quién tiene la responsabilidad última de la publicación del material. Algunas de las preguntas que se plantean para aplicar este criterio son las siguientes:
  - ¿Quién es el autor del documento o del sitio web?
  - ¿Se dan sus afiliaciones y cualificaciones?
  - ¿Hay información de contacto: dirección, número de teléfono, dirección de correo electrónico?
  - ¿Cuál es la organización que publica o patrocina? ¿Hay un enlace a su página principal?
  - ¿Es una página personal o institucional?

7.5.2. El contenido

El **contenido** es el punto principal en la evaluación de los recursos. En la política de INTUTE se indica que el interés principal radica en la información contenida en el recurso. La calidad del diseño o la apariencia tienen un interés secundario, aunque pueden afectar a la utilidad de los recursos.

Web recomendada

The Intute Consortium, http://www.intute.ac.uk/about.html

- **Rigor en el tratamiento de la información**. Este criterio hace referencia a que la información es fiable y que no hay errores. También es uno de los criterios aplicados tradicionalmente a los documentos impresos.
  Debemos observar con detenimiento los siguientes puntos:
  - ¿Es la información válida, fiable?
  - ¿Se utiliza una ortografía y una gramática correctas?
  - ¿Están los datos comprobados?
  - ¿Se documentan las fuentes de información utilizadas?
  - ¿Hay enlaces a esas fuentes de información?
  - ¿Es verificable?

- **Exhaustividad, volumen de la información**. En las fuentes impresas incluyen frecuentemente un prefacio o introducción al principio de la publicación, explicando los temas tratados en el trabajo, la profundidad o el nivel con el que se estudian y el público al que van dirigidos.
  La **cobertura** de los recursos en la Web puede ofrecernos más complicaciones. En las páginas o en los sitios web suele faltar el equivalente a un pre-

facio o introducción; la cobertura no se aprecia tan rápidamente. Recorrer totalmente un sitio web puede ser molesto, por lo que sólo la existencia de un **índice o un mapa del sitio** en el que se incluyan todos los temas tratados y la profundidad del estudio pueden ayudarnos.

- **Actualización**. Expresa el grado en el que un material está al día. Ha sido considerado tradicionalmente como uno de los criterios para evaluar fuentes de información. Como no hay directrices establecidas para incluir las fechas en las páginas web, puede ser difícil determinar la actualidad de las fuentes de la Red. Frecuentemente, las fechas de publicación no aparecen o, si aparecen, se confunde fácilmente la fecha en la que se creó el documento, la fecha que se puso en la web y la fecha en la que ésta se modificó por última vez.

- **Objetividad**. Es importante intentar asegurar la objetividad del proveedor de información. Si estamos familiarizados con la fuente de información en la Red, evaluar su objetividad no presentará probablemente más dificultades de las que ofrecen los materiales impresos. Pero puede ser difícil determinar la objetividad de una página web, a menos que el objetivo del individuo o grupo que presenta la información esté claramente establecido.

- **Originalidad**. Si el recurso nos ofrece una información que sólo podemos conseguir mediante su consulta.

7.5.3. El acceso y la interactividad

Están incluidos en este grupo:

1) Descarga y condiciones de acceso

Algunos autores entienden que éste debe ser el primer criterio que hay que tener en cuenta, ya que antes de juzgar la calidad de un recurso es necesario localizarlo y acceder al servidor en el que se encuentra el documento. Son aspectos relacionados con la primera impresión que reciben los usuarios del sitio web: facilidad de conexión y de descarga, identificación del sitio, restricciones de acceso, accesibilidad y capacidad de uso, política de cobro, requisitos especiales, copyright y otras cuestiones que se deben resolver antes de que se pueda usar la información contenida en el sitio.

Web recomendada

Evaluating the Quality of Internet Information Sources. http://www.eric.ed.gov/ERICWebPortal/custom/portlets/recordDetails/detail mini.jsp?_nfpb=true& _&ERICExtSearch _SearchValue_0=ED412927&ERICExtSearch_Search Type_0=no&accno=ED412927

2) Capacidad de recuperación y facilidad de uso

Si el recurso que estamos seleccionando y evaluando ha de ser utilizado por unos usuarios concretos, debemos tener muy presentes cuáles serán sus conocimientos y sus capacidades técnicas para establecer si serán capaces de extraer el máximo rendimiento al recurso en concreto, por lo que deberemos fijarnos en los siguientes puntos:

La usabilidad y la accesibilidad

La usabilidad y la accesibilidad de un recurso están afectadas por las características de Internet como medio de publicación. Se trata de unos rasgos que no existían en el mundo impreso.

- **Servicios de recuperación de la información**: ¿Es posible hacer búsquedas por palabras o frases?, ¿se pueden utilizar operadores booleanos?, ¿se pueden utilizar otros operadores: proximidad, comparación, etc.?

- **Ayudas**: ¿Existen ayudas para la recuperación y los sistemas de recuperación de información adicionales (tesauros, listas de temas, etc.)?

Ved también

Podéis ver el apartado 8, "Arquitectura de la información".

3) Navegación y representación de la información

La **navegación** se refiere a la facilidad con la que los documentos pueden ser explorados y está relacionada con la arquitectura de la información.

Información coherente

Es imprescindible una coherencia en la navegación, es decir, un estilo informativo coherente que afecte a: colores, iconos y tipos de menús.

Es conveniente que el recurso cuente con un **sumario general** y que éste se encuentre en la primera sección de la web. Se considera básico que desde cualquier nodo se vaya al sumario principal con un simple clic y que desde cualquier nodo se vaya a cualquiera de las secciones principales.

Ved también

Podéis ver el apartado 8, "Arquitectura de la información".

4) Interactividad

Aparte de la capacidad de recuperación, considerada por Codina como la primera forma de interactividad, a un primer nivel, hemos de comprobar si se pueden plantear preguntas, sugerencias al administrador web. Pero también hemos de buscar si se ofrece algún otro modo de interactividad, como el uso de formularios.

5) Enlaces

Uno de los aspectos que distinguen a los recursos de Internet basados en el hipertexto es la habilidad de enlazar un documento con materiales o recursos relacionados.

La **calidad** de los enlaces se refiere a su utilidad y a la claridad con la que están descritos. Es necesario distinguir si ofrecen una simple lista de enlaces a otras sedes o una verdadera selección de éstos, descritos y actualizados, y si en esa lista las propias sedes enlazadas han sido evaluadas.

6) Visibilidad (popularidad, impacto)

Uno de los criterios que propone Codina, y que no suele aparecer en otras propuestas, es la **visibilidad**. Ésta se mide por el número de otras páginas web que mantienen enlaces con la web objeto de análisis, utilizado en ocasiones como un criterio de calidad.

El hecho de que otros autores o instituciones enlacen sus páginas web con una página determinada puede hacernos pensar, lógicamente, que la han considerado de cierta calidad. Los enlaces han sido equiparados a las citas bibliográficas recibidas y han centrado una corriente de estudios que pretende aplicar las técnicas de la bibliometría tradicional a la nueva situación creada por Internet.

7) Autodescripción

El último criterio de este grupo se refiere a la descripción que se hace en una página de ella. Estos datos identificativos serán los que los motores de búsqueda "lean" sobre la página y los que se tendrán en cuenta a la hora de ofrecer resultados para las cuestiones que se les planteen.

Por lo tanto, es necesario ser cuidadoso en la fase del diseño e intentar dar una información completa y acertada en estos campos. Las páginas web que están bien autodescritas son más accesibles a través de los motores de búsqueda y las agencias de evaluación, por lo que es más fácil que sean localizadas y utilizadas. Para que una página web esté bien autodescrita debe tener:

- Título: un título transparente.
- Contenido: los primeros párrafos de la web y los primeros nodos contienen texto, no gráficos, en el que se hace explícito el contenido.
- Las metaetiquetas se ubican en la sección HEAD de un documento HTML. La inclusión de estas metaetiquetas permite al motor de búsqueda localizar rápidamente los recursos que contienen la ecuación de búsqueda.

Web recomendada

Metadatos DCMI Home: Dublin Core Metadata Initiative. http://dublincore.org/

7.5.4. El diseño y la estética (legibilidad y ergonomía)

Los criterios de diseño están muy relacionados con la navegabilidad. Se trata de dos grupos de criterios que no tenían apenas significado en la evaluación de fuentes impresas. Internet ha requerido nuevos indicadores para analizar unas características que antes no existían.

Gene Wilkinson, Lisa T. Bennett y Kevin M. Oliver opinan que la capacidad de uso de la información depende en cómo se organice ésta. Dentro de la estructura de la información y el diseño, se contemplan puntos como la estructura del documento y si éste sigue unas normas estandarizadas de diseño tales como establecer los objetivos, describir su alcance, incorporar interactividad o proporcionar una variedad de formatos.

Web recomendada

Evaluando la calidad de las fuentes de información del Internet: listado consolidado de criterios de evaluación e indicadores de calidad. http://www.sg.inter.edu/lisc/pub/criterios.html

Así, de manera general, se destaca que:

- El **diseño** debe ser intuitivo con unos elementos icónicos claros.
- Las páginas largas o complicadas se deben dividir en archivos más pequeños.
- Las **imágenes** no deben servir sólo como pura decoración, sino que deben proporcionar información.
- Todas las páginas deben tener un título corto, informativo, descriptivo, con una **tipografía** adecuada que facilite la **lectura** de la información.
- Todas las páginas deben tener (conviene incluirlo ya en la plantilla): la fecha de la última actualización, una dirección de correo electrónico para contactar con la persona responsable del contenido de la página y un enlace a la correspondiente página principal.

Ved también

Podéis ver el apartado 8, "Arquitectura de la información".

7.6. La catalogación de los recursos

Una vez que se decide evaluar un recurso, se procede a la elaboración de una **ficha descriptiva**. La descripción de cada recurso sirve para asesorar rápidamente a los usuarios del origen, el contenido y la naturaleza, lo que permite a los usuarios decidir sobre el valor de cada recurso para su investigación.

Las **agencias de evaluación** se caracterizan por la creación de descripciones individuales de los recursos de Internet incorporados a su base de datos, que constan de una serie de diferentes atributos como título, autor, URL, materia, etc.

Modelos de plantilla

Codina sugiere un modelo de ficha descriptiva que incluye tres modelos de plantilla, uno para la identificación, otro para el análisis, anotación y tabulación, y un tercero para las conclusiones (http://www.lluiscodina.com).

7.7. Herramientas de análisis, evaluación y validación automáticas

Los navegadores son muy resistentes a los errores en el código HTML. La ventaja es que, a diferencia del código de un programa informático, las páginas web podrán visualizarse a pesar de una mala codificación. Indirectamente se está propiciando la creación de páginas con errores de código HTML, ya que los creadores de páginas web no se ven obligados a ser cuidadosos en su trabajo.

Web recomendada

Herramientas de validación W3C. http://www.w3.org/Status, http://www.w3.org/QA/Tools/

Esta deficiente codificación no tendría mayores consecuencias si la Web no evolucionara y no hubiera desarrollos previstos más potentes que exijan una codificación impecable, como la web semántica y la posibilidad de un uso mucho más "inteligente".

Web recomendada

Guía breve de la web semántica (W3C). http://www.w3c.es/Divulgacion/Guiasbreves/WebSemantica

Web usable, accesible y localizable

Por otro lado, una página web, además de estar bien codificada, es conveniente que sea usable, accesible y fácilmente localizable, ya que difícilmente podrá cumplir con sus ob-

jetivos si no puede ser localizada en un buscador o no puede visualizarse correctamente por personas con algún tipo de discapacidad.

Ved también

Podéis ver el apartado 8, "Arquitectura de la información".

7.8. Plantillas y métodos de evaluación

Primera lista de criterios aparecida en España, propuesta por Lluís Codina y respaldada por el máster de Documentación digital de la Universidad Pompeu Fabra. Contiene métodos, conceptos, procedimientos y plantillas de análisis de sitios web y/o de publicaciones digitales. Todo el material es de dominio público y pueden ser libremente descargado.

Web recomendada

Ll. Codina. "Evaluación de recursos digitales". http://www.lluiscodina.com

El artículo propone la elaboración de un cuestionario de evaluación para sitios web. Mediante el estudio de la bibliografía especializada, analiza los criterios en los que se basa y expone la metodología; a continuación, describe un cuestionario, compuesto por 8 bloques, divididos a su vez en 16 criterios y 125 cuestiones, al que se asigna una puntuación ponderada; finalmente, expone las conclusiones obtenidas. El trabajo se complementa con un anexo que incluye el propio cuestionario y las tablas de datos utilizadas en su elaboración.

Web recomendada

M. Jiménez Piano. "Evaluación de sitios web". *Revista española de Documentación Científica* (24, 4, 200). http://redc.revistas.csic.es/index.php/redc/issue/view/7

Recopilación de métodos de evaluación de recursos en Internet

Web recomendada

The World-Wide Web Virtual Library Evaluation of information sources. http://www.vuw.ac.nz/staff/alastair_smith/evaln/evaln.htm

Web recomendada

M.ª D. Ayuso García; V. Martínez Navarro (2005). "Protocolo de evaluación de fuentes y recursos informativos en la sociedad del conocimiento: propuestas, enfoques y tendencias". *Revista General de Información y Documentación* (15, núm. 1, pág. 21-53). http://revistas.ucm.es/portal/modulos.php?name=Revistas2_Historico&id=RGID&num=RGID050512

Este trabajo presenta un estudio de las propuestas de evaluación de fuentes de información publicadas en Internet y llevadas a cabo tanto por estudiosos de la materia como por instituciones científicas y académicas. Contiene un protocolo de evaluación de propuesta propia.

7.9. Las agencias de evaluación: definición, finalidades y objetivos

Las **agencias de evaluación**, tanto especializadas en un área concreta del conocimiento como de ámbito más general, ofrecen a los usuarios el acceso a una base de datos de descripciones de recursos de Internet en la que pueden tener la certeza del fuerte control de la calidad de los recursos, lo que constituye el producto por excelencia dentro de la evaluación de recursos digitales en línea.

Podemos decir que las agencias de evaluación trabajan siguiendo un principio y seleccionan, clasifican y catalogan recursos de Internet con la finalidad de ayudar en la búsqueda y recuperación de la información a sus usuarios.

Son un servicio de calidad controlada que reúne las siguientes características:

- Servicio en línea que ofrece enlaces a numerosos sitios web o documentos en Internet (recursos).
- En primer lugar, elabora una lista de criterios, establece una escala de calidad y fija una metodología de evaluación.
- Los recursos pasan por un cuidadoso proceso de selección antes de ser incluidos en el directorio. Esta selección es un proceso intelectual que se realiza manualmente, siguiendo unas normas de calidad preestablecidas y publicadas.
- Se realiza una descripción de contenidos, desde una breve reseña hasta una referencia analítica. Esto excluye automáticamente los denominados sumarios. No necesariamente se deben asignar palabras clave o términos controlados, aunque es muy recomendable.
- Tiene una estructura o clasificación establecida, por lo que se excluyen las listas de términos sin estructuración.
- Se les asignan manualmente metadatos para cada recurso individual.

Por lo tanto, podemos decir que una agencia de evaluación es un organismo, del sector público o privado, que se encarga de la búsqueda, selección y descripción de recursos digitales siguiendo una metodología y unos criterios establecidos de selección y evaluación basados en la calidad del recurso.

Ejemplo

INTUTE: http://www.intute.ac.uk/

El Proyecto Intute, de consulta gratuita, fue creado por un consorcio de siete universidades del Reino Unido y sociedades científicas. Financiado por Joint Information Systems Committee (JISC) (http://www.jisc.ac.uk/), es un directorio selectivo de recursos web en línea realizado por una red de especialistas en las distintas materias que clasifica y describe recursos de calidad disponibles en Internet y que está dirigido a la enseñanza e investigación académica.

8. Arquitectura de la información en sitios web

8.1. Introducción

La **arquitectura de la información** (AI) tiene como objetivo el diseño estructural de los sistemas de información, especialmente los que se implementan con tecnología web. Comprende la organización de la información, su rotulación, el diseño de la interacción, el diseño de navegación y el diseño de la interfaz de usuario (Rovira, 2003).

Lectura recomendada

C. Rovira (2003). "Arquitectura de la información en sedes web" [en línea]. En Cristòfol Rovira; Lluís Codina (dirs.). *Documentación digital*. Barcelona: Sección Científica de Ciencias de la Documentación, Departamento de Ciencias Políticas y Sociales, Universidad Pompeu Fabra. http://www.documentaciondigital.org. ISBN 84-88042-39-6

El término fue acuñado por **Richard Saul Wurman** en 1975, quien la define como:

> "El arte y la ciencia de estructurar y clasificar la información con el objetivo de permitir al usuario encontrar su vía de navegación hacia el conocimiento y la comprensión de la información".

Aunque hasta 1998 Rosenfeld y Morville no lo popularizaron.

Lectura recomendada

R. S. Wurman; P. Bradford (eds.) (1996). *Information Architects*. Zúrich: Graphis Press. ISBN:3-85709-458-3.

Arquitectura de la información

Al igual que la usabilidad, como veremos más adelante, mantiene una relación con la psicología, la interacción persona-ordenador o la sociología. Respecto a las ciencias de la información y la documentación, la AI mantiene una relación directa, entre otras, con la clasificación y catalogación de la información, descripción o metadatos, los vocabularios controlados, la recuperación y visualización de información y los estudios de necesidades de información de usuarios.

Lectura recomendada

L. Rosenfeld; P. Morville (2002). *Information Architecture for the World Wide Web. Designing Large-Scale Web Sites*. California: O'Reilly.

R. Ronda León. "Arquitectura de Información: análisis histórico-conceptual y La Arquitectura de la Información y las Ciencias de la Información". En: *No Solo Usabilidad*. ISSN: 1886-8592. http://www.nosolousabilidad.com/articulos/ai_cc_informacion.htmhttp://www.nosolousabilidad.com/articulos/historia_arquitectura_informacion.htm

8.2. Objetivos

Los objetivos de este apartado se centran en analizar las características de la navegación hipertextual y los sistemas de navegación eficientes, así como en ofrecer los elementos necesarios para el diseño de una estructural eficaz, usable y accesible de sistemas de información, especialmente los que se implementan con tecnología web.

8.3. Navegación hipertextual

La World Wide Web es un servicio de Internet que utiliza la navegación hipertextual para acceder de un modo rápido a la información, lo que forma una gigantesca red de documentos interrelacionados.

Se denomina **navegación** a la acción de consultar un hipertexto activando sus enlaces. La navegación por un hipertexto es, por lo tanto, un proceso de observación y manipulación en la pantalla para ir explorando espacios informativos, mientras se configura como un medio para realizar una tarea, que es adquirir información y conocimientos.

Codina habla de sitio web o sede web, tomándolos como sinónimos, para referirse a una localización en la World Wide Web, que se identifica por una URL que almacena una o varias páginas web, unidas por enlaces hipertextuales para facilitar su acceso y su utilización.

Lectura recomendada

Ll. Codina Bonilla (2000). "Evaluación de recursos digitales en línea: conceptos, indicadores y métodos". *Revista Española de Documentación Científica* (vol. 23, núm. 1, pág. 9-44). http://www.lluiscodina.com/metodos.htm

Por lo tanto:

Una **sede web** es la unidad virtual, identificada por un URL, que almacena páginas web para su consulta.

Una de las ventajas de la información digital es que permite su actualización de una manera relativamente poco complicada técnicamente, económicamente rentable e instantánea a través de las redes telemáticas.

8.4. Hipertexto

Gracias al **hipertexto**, podemos decir que las páginas de los documentos digitales son "navegables", el usuario puede elegir su propio itinerario o recorrido de lectura al enfrentarse al documento.

Por lo tanto:

Un **hipertexto** es un documento digital que aprovecha la ventaja de la computabilidad para permitir un acceso asociativo a la información, rompiendo con la secuencialidad de la información en papel.

(Rovira, 1998)

Lectura recomendada

C. Rovira Fontanals (1998). "L'hipertext: la recuperació d'informació per navegació al web". En: J. Baró i Queralt. *Cercar i col·locar informació en el World Wide Web* (pág. 57-79). Barcelona: Llibres de l'Index.

La información en un hipertexto se organiza en unos bloques de contenido denominados **nodos**, que se conectan entre ellos mediante enlaces. Los **anclajes** son los puntos físicos de salida y llegada de un enlace. Al pulsar un anclaje,

se activa el enlace y se produce un **salto hipertextual** que nos lleva del nodo o documento que estábamos consultando a otro que mantiene una relación asociativa con el primero.

Conklin propone una serie de características que definen un hipertexto ideal:

- Una base de datos en red formada por nodos de información textual y gráfica.
- Los nodos de la base de datos se visualizan en la pantalla del ordenador por medio de ventanas. Una ventana corresponde a un nodo y sólo se pueden ver un pequeño número de ventanas al mismo tiempo en pantalla.
- Las ventanas se utilizan siguiendo las convenciones estandarizadas (abrir, cerrar, etc.).
- Las ventanas contienen enlaces que representan conexiones a otros nodos de la base de datos. Los enlaces contienen texto para explicitar el contenido del nodo apuntado, y la acción de activarlo hace que se abra una ventana y se muestre su contenido.

Lecturas recomendadas

J. Conklin (1987). *Hypertext: an introduction and survey*. *Computer* (sept., vol. 20, núm. 9, pág. 17-41).

Ll. Codina (1997-1998). "H de Hypertext, o la teoría de los hipertextos revisitada". *Cuadernos de Documentación Multimedia* (núm. 6-7). Disponible en: http://www.ucm.es/info/multidoc/multidoc/revista/cuad6-7/codina.htm

V. Bush. "As we may think". http://www.w3.org/History/1945/vbush/

8.5. Los nodos

Los **nodos** son las unidades básicas del hipertexto. Pueden estar formados por lo que consideraríamos un capítulo, una sección o uno o varios párrafos, una pantalla o un documento completo.

La forma y las dimensiones de un nodo son completamente arbitrarias, sino que depende de la estructura que le dé el autor o de las características propias de la información contenida estructurada por el hipertexto.

8.5.1. El tamaño de los nodos

Las teorías clásicas del hipertexto de Bush, Conklin, Engelbart, Nelson y Landow sugieren como extensión óptima la del desarrollo de **una sola idea o concepto**, es decir, la creación de nodos pequeños. Se basan en que la riqueza del hipertexto se encuentra en potenciar al máximo la libertad del lector y que éste sea quien vaya construyendo el discurso a través de la navegación por los documentos hipertextuales.

Para ello, los nodos deben desarrollar una sola idea e incluir enlaces asociativos a otros nodos que desarrollen ideas relacionadas (Rovira, 1998).

La mayor ventaja que presentan los **nodos pequeños** es la flexibilidad en la adaptación de los contenidos desde el punto de vista de los conocimientos previos del lector. La navegación permite evitar aquellos nodos que contienen información no pertinente o conocida ya por el lector.

Los inconvenientes que puede plantear esta argumentación es que actualmente en la Web es difícil establecer *a priori* si un nodo contiene información relevante o no, aunque intentan establecerse instrumentos que lo faciliten. Cuanto más pequeño es un nodo, la información resulta más fraccionada, inconexa y dependiente de los enlaces. Otro inconveniente que presentan los nodos pequeños es que propician aún más la sensación de desbordamiento cognitivo porque el número de saltos hipertextuales puede ser grande.

Los **nodos grandes** presentan como ventajas una mayor economía de medios y una menor sensación de desorientación por parte del lector. Sin embargo, queda reducida la sensación de libertad y es menor la adaptación a los conocimientos previos del lector.

En el caso de documentos largos, el enlace hipertextual interno permite saltar desde un punto a otro de la misma página y se puede acceder a cualquier parte del documento por medio de sumarios, etc. Es un recurso muy utilizado con la finalidad de evitar la utilización del *scroll* para volver al inicio del texto.

Jakob Nielsen

Como afirma Jakob Nielsen, sólo el 16% de los lectores de documentos digitales leen palabra por palabra, el resto simplemente las hojean. Propone una serie de medidas, como poner encabezados significativos, listas con viñetas, una idea por párrafo, empezar el texto con las conclusiones y utilizar la mitad o menos del texto que se utilizaría en un texto convencional.

Para evitar el **desbordamiento cognitivo** es necesario un diseño de hipertexto proporcionado, dividido en nodos de tamaño conveniente, con enlaces significativos que ofrezcan una idea completa, adaptando la información a las previsiones de capacidad de lectura del lector, suministrando herramientas de ayuda a la navegación en un entorno de diseño agradable.

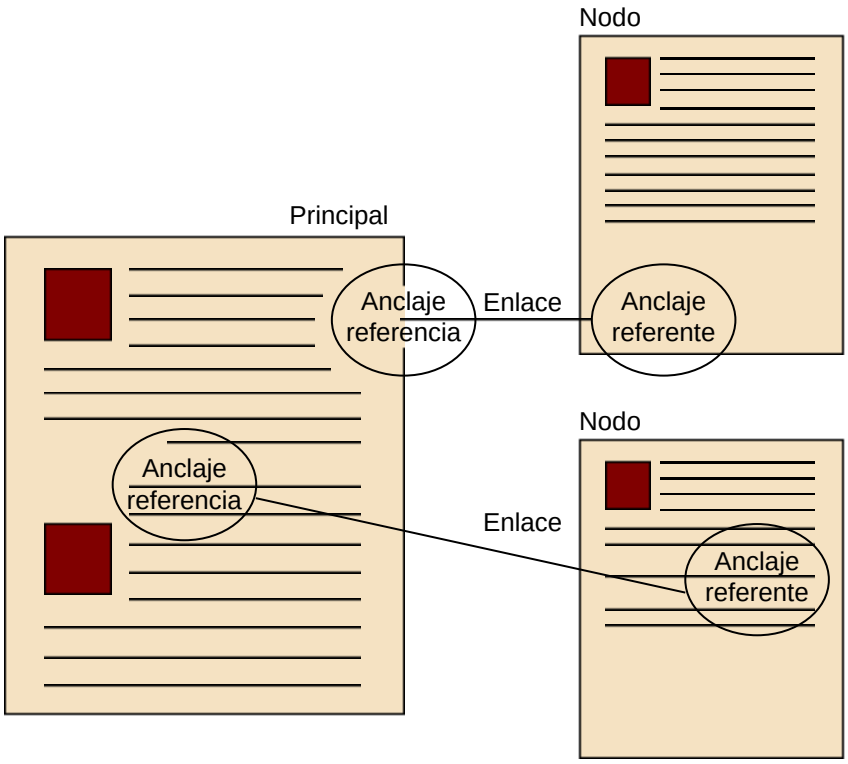
El **desbordamiento cognitivo** es el efecto de desorientación en el lector debido a la ausencia de herramientas de navegación y de una estructura lógica y clara en los hipertextos.

8.6. Los enlaces

Un **enlace** es una conexión entre dos nodos que proporciona un modo de seguir las referencias entre un origen y un destino.

Los extremos del enlace se denominan **anclajes**.

El anclaje origen se conoce como **referencia** y el de destino, como **referente**.



La activación de un enlace puede dar lugar a diferentes resultados (texto, imagen, sonido, definición, información adicional, nueva sede web, etc.), pero siempre une dos nodos vinculados semánticamente.

Los enlaces se pueden clasificar de distintas maneras. La más básica los clasifica en enlaces hipertextuales **externos** y enlaces hipertextuales **internos**.

- El **enlace hipertextual externo** une dos nodos situados en archivos diferentes.
- El **enlace hipertextual interno** une dos puntos del mismo nodo o página.

Según su organización (Rovira, 1997) tenemos:

- Enlaces **jerárquicos**. Unen los nodos en función del grado de especificidad del tema tratado. Habitualmente se reflejan en un menú o sumario con estructura jerarquizada.
- Enlaces **asociativos**. Unen los nodos por una proximidad de ideas y forman una estructura en forma de red.
- Enlaces **secuenciales**. Conecta los nodos de manera lineal, de manera que cada nodo tiene un enlace que lo lleva al nodo anterior de la ruta establecida y otro que lo lleva al nodo posterior.
- Enlaces de una **estructura secuencial**. Una estructura secuencial de enlaces se utiliza para establecer visitas guiadas, unir nodos consecutivos y evitar páginas demasiado largas, así como permitir una consulta secuencial del documento.

Lectura recomendada

C. Rovira Fontanals (1997). "Entornos hipertextuales de aprendizaje". En J. Baró i Queralt; P. Cid Leal (eds.). *Anuario SOCADI de Documentación e Información* (pág. 121-127). Barcelona: Societat Catalana de Documentació i Informació. Disponible en: http://www.raco.cat/index.php/Bibliodoc/article/viewFile/56354/65776

8.7. Principios de diseño y de navegación

8.7.1. Estrategias de navegación

La **navegación** se considera un proceso de recuperación de información en el que se parte de una demanda de información y se obtiene un conjunto de documentos que la satisfacen.

La **recuperación por navegación** se limita a las propiedades asociativas expresadas por los enlaces.

Existen dos estrategias básicas en la navegación hipertextual (Rovira, 1998):

- **Navegación en amplitud**. Consiste en ir activando y retrocediendo por todos los enlaces del nodo activo para visitar todos los nodos vecinos antes de avanzar a un nuevo nodo y proseguir del mismo modo.

- **Navegación en profundidad**. Se avanza siempre más allá del nodo activo eligiendo alguno de los posibles enlaces y sólo retrocediendo cuando se llega a un nodo sin enlaces.

Entre estas dos estrategias de navegación existen otras intermedias. Se recomienda combinar ambas en función de los resultados obtenidos en cada momento, según la adecuación del enlace al tema. Navegaremos en profundidad si este está tratado de una manera un tanto próxima y navegaremos en amplitud si el tema es tratado en plenitud.

8.7.2. El proceso de diseño de una sede web

El diseño de una página web no es solamente un problema estético, también intervienen otros factores, como la organización de la información, la claridad y la comodidad de utilización, la velocidad de acceso y las posibilidades de interacción.

Las ocho cualidades que debe reunir una sede web eficaz son:

1) Rapidez
2) Accesibilidad
3) Estabilidad
4) Evolución
5) Interactividad
6) Actualidad
7) Seguridad
8) Conocimiento

El proceso de diseño ha de seguir una serie de etapas (Lynch, 2008):

- El primer paso que debe darse a la hora de diseñar una página web es definir con claridad los **objetivos** que va a cumplir y la finalidad que va a tener.

- Los objetivos se deben establecer de manera clara y concisa, y el diseño ha de estar en función de la **satisfacción de las necesidades de los usuarios** y de las características de éstos. Se ha de adoptar siempre sus puntos de vista e imaginar las sensaciones y el flujo de sus pensamientos.

- Es muy importante tener una idea clara de cuáles van a ser los **contenidos** de la página antes de comenzar con el proceso de diseño. Para esto, resulta muy útil disponer de un **inventario** de la información de la que disponemos o queremos incorporar a la web.

- **Definición** de la sede web y **planificación**. Se deben definir los objetivos, el presupuesto, el alcance de los contenidos, la interactividad, el soporte tecnológico y la cantidad y calidad de los recursos de información necesarios para satisfacer a los usuarios.

- Es importante fijar cuál es el **propósito** de los autores de la página, en qué puede ayudar la estructura de la web a satisfacer estos propósitos, cuáles son los objetivos a corto y largo plazo, qué estrategias se deberán poner en práctica en la página para cumplir los objetivos y cómo se medirá el éxito o el fracaso en el cumplimiento de los objetivos.

- Establecer la **arquitectura de la información**. Se definirán los contenidos y la organización de la sede. Es conveniente inventariar la información disponible y ver la información nueva que es necesaria, así como definir la estructura organizativa de la sede. En este paso se elaboran los **prototipos** con unas cuantas páginas.

- Fijar el **diseño** de la sede. Se elabora el aspecto y el comportamiento final del producto. Se crearán las plantillas de las páginas, el diseño de éstas y el de los gráficos de uso general. Se maquetarán los materiales informativos (imágenes, textos, bases de datos, etc.) y se pondrán en funcionamiento preparándolos para el montaje final.

- **Construcción** de las páginas. Se generarán las páginas, se refinarán aún más los diseños y se realizarán las primeras pruebas con los usuarios.

- **Difundir** la sede web. Dependerá del público al que ésta se dirige. Es conveniente dar la dirección de alta en buscadores de tipo general. La visibilidad de la sede web dependerá de dónde se halla referenciada. Es lo que

Lectura recomendada

P. J. Lynch; S. Horton (2008). *Web stile guide: basic design principles for creating web sites*. New Haven: Yale University Press.

se denomina SEO, sigla en inglés de *search engine optimization*, u optimización para motores de búsqueda.

- Ponerla en marcha, establecer un proceso de **evaluación** y el **mantenimiento**. Se pueden obtener datos de las visitas que recibe, qué páginas se visitan más o cuánto tiempo permanecen en ellas los usuarios, o de qué país proceden. Esto nos da información para saber qué páginas están más saturadas y para detectar errores en la navegación. El mantenimiento debe atender las consultas de los usuarios y actualizar los contenidos y enlaces.

"Es importante leer el documento adaptándolo al mayor número de navegadores y supervisar la legibilidad del documento. También se deben aceptar las críticas o sugerencias de los usuarios lectores, así como preguntarles sobre los problemas que encuentran en el documento, y analizar sus causas".

Lecturas recomendadas

T. Berners-Lee (1992-1998). "Style guide for online hipertexto". Disponible en: http://www.w3.org/Provider/Style/All.html

"El ciclo de vida de una campaña de posicionamiento. Fundamentos del marketing en buscadores. Vigilancia tecnológica e inteligencia competitiva para SEM-SEO". En C. Rovira; Ll. Codina; M.-C. Marcos; R. Pedraza (2008). Máster Online en Buscadores. Selección de unidades didácticas 2007/2008 (1.ª ed., septiembre). Disponible en: http://www.lluiscodina.com/

8.8. La estructura de la información

Es fundamental tener en cuenta las características de este medio y estructurar la información de acuerdo con esto y con las necesidades de los usuarios lectores.

Una página web se estructura en tres partes:

- La **cabecera**. Contiene el título del documento, el enlace a la página de inicio (si no lo es), la institución responsable, etc.

- El **cuerpo**. Desarrolla el contenido del mensaje que quiere transmitir la página.

- El **pie**. En él se sitúa la información sobre la autoría, cómo contactar con la persona o institución responsable, la fecha de creación o revisión y los enlaces a la página de inicio y a otras dentro de la misma sede web. Es también conveniente incluir la información acerca de la legislación o convención que ampara la utilización o reproducción de la página.

Cabecera

Cuerpo

Pie

La **página principal** funciona como entrada lógica al sistema y todas las demás páginas se organizan en torno a ella.

El **contenido de la información** de una página debe seguir los principios del periodismo:

- Debe reflejar **quién** es el autor y, teniendo en cuenta que la web es un medio bidireccional, los usuarios deberían poder enviar comentarios o sugerencias a los autores, por lo que conviene ofrecer opciones para contactar.
- Debe tener **títulos** visibles que capturen la atención del lector. Es importante tener en cuenta que el título es lo primero que se ve en una página cuando se está cargando, es el texto que aparece en la lista de marcadores que almacena un usuario (*bookmark*) y es la información más significativa que recogen los motores de búsqueda.
- Es muy recomendable incorporar la **dirección de la página principal** en todas las páginas, ya que el usuario puede acceder a una página sin haber entrado en la página principal, por ejemplo mediante los resultados a una consulta en un motor de búsqueda, con lo que se encontraría con una página fuera de contexto.

Importancia de la estructura de una página web

El área más visible de una página es la **parte superior** que se visualiza en un monitor. Esta zona debería presentar el mayor número de enlaces posible; la zona inferior, el **pie**, también es útil e importante, pero no tiene un impacto tan directo en los lectores.

Se recomienda que sea una página corta y simple, ya que la simplicidad hace más fácil su uso. La página principal es la más visitada y el mejor lugar para exponer noticias e información, así como el menú de enlaces o la tabla de contenidos.

Lectura recomendada

J. Nielsen (1996). "Inverted pyramids in ciberspace". *Alertbox* (junio). Disponible en: http://www.useit.com/alertbox/9606.html

Nielsen aboga por organizar la información siguiendo el modelo de **pirámide invertida**, teniendo en cuenta que los usuarios en la Web fundamentalmente recorren con la vista los documentos web en lugar de realizar una lectura detenida.

Este modelo es muy utilizado en los medios periodísticos y consiste en empezar a redactar colocando la conclusión en primer lugar, luego proseguir con la información más importante que sustente esta conclusión y finalizar con los aspectos de fondo del tema.

En la Web este modelo cada vez es más utilizado, puesto que los usuarios prefieren no utilizar el *scroll* o desplazamiento de pantalla, y muy frecuentemente leen solamente la parte superior de un documento.

Siguiendo a Nielsen se recomienda:

- Organizar cuidadosamente la información.
- Empezar cada párrafo con una frase que resuma el tema.
- Utilizar palabras y categorías que la audiencia pueda entender.
- Limitar en cada párrafo una idea principal.
- Colocar la cantidad justa de información.
- Los textos deben ser cortos.
- Los subencabezamientos deben estar llenos de significado.
- Los elementos gráficos deben complementar el texto.
- Eliminar adjetivos y todo aquello que recuerde a un lenguaje "promocional".
- Escribir un 50% del texto que se usaría en un formato tradicional para eliminar el *scroll*.
- Destacar y enfatizar las palabras que debe capturar el ojo del usuario.
- Estructurar el texto en múltiples páginas y unirlas con enlaces, relegando la información menos importante a páginas secundarias (jerarquía) y estructurar el texto en dos o como mucho tres niveles de encabezamientos.

Sugerencias de diferentes autores

Nielsen sugiere reescribir las sedes web eliminando casi la mitad de información para mejorar su capacidad de uso y que no se debe utilizar la última tecnología en la confección de páginas web, ya que al más mínimo error el usuario abandonará la sede web inmediatamente con sensación de frustración.

Otra de las recomendaciones interesantes es la de escribir un hipertexto que sea imprimible porque en el mundo real muchos documentos digitales generan una copia en papel. Es posible generar una versión imprimible elaborando un enlace que nos lleve a un único documento y secuencial. T.Berners-Lee (1992-1998). *Style guide for online hypertext* . Disponible en: http://www.w3.org/Provider/Style/All.html

En un estudio realizado por Nielsen y Morkes se demostró que a los usuarios al entrar en una sede web les gustaría encontrar una opción de búsqueda; recomiendan gestionar mejor el interior de una sede web, con mapas de contenido y sumarios. Las facilidades de búsqueda son imprescindibles en páginas web grandes y resultan convenientes en las pequeñas si contienen documentos largos.

Frequently asked questions

Un tipo de ayuda para encontrar información de un modo rápido y que se ha popularizado en la Web son las **FAQ** (*frequently asked questions*), una lista de preguntas comunes o frecuentes sobre un tema, acompañadas de una respuesta clara y concisa.

Eficacia de enlaces

Cornellà destaca que el diseño gráfico no parece influenciar en el éxito de la búsqueda de información, que los enlaces deben indicar claramente lo que se consigue al pulsar sobre ellos y que los usuarios no parecen construir un modelo mental de la Web, simplemente navegan hasta que encuentran lo que buscan y hay que ayudarles a encontrar lo que necesitan.

Lectura recomendada

A. Cornellà Solans (2000). "Diseño de páginas web y búsqueda de información". *El profesional de la información* (vol. 9, núm. 3, marzo). Disponible en: http://www.elprofesionaldelainformacion.com/ contenidos/2000/marzo/1.pdf

8.9. Los elementos gráficos

Los usuarios encuentran poco placentero el hecho de esperar, y texto e imagen forman una combinación poderosa que debe complementarse, no ser una distracción ni una pérdida de tiempo, por lo que se considera importante la **integración** de la información de los gráficos con el texto.

La Web es un medio altamente visual y se debe evitar sobrecargar con grandes gráficos o provocar confusión a causa de una profusión de gráficos pequeños.

Patrick Lynch y Sarah Horton recomiendan la utilización de plantillas de diseño que aporten consistencia y predictibilidad a las páginas. Consiste en predefinir una plantilla de composición, con un estilo formal que se aplique a todas las páginas de una sede web y les dé un ritmo de unidad. La repetición no aburrirá al lector y proporcionará una identidad gráfica consistente. Cada sede web debe aplicar una plantilla propia, ya que ningún sistema de plantillas es adecuado para todas las sedes web.

Dado que la fuente tipográfica ha de estar disponible en el ordenador del usuario, es preferible utilizar tipos de letra estándar:

- La **cursiva** se utiliza solamente en los casos que establece la normativa: en títulos de libros y publicaciones periódicas, extranjerismos, palabras que destacar en un texto, etc.
- La **negrita** es adecuada para subtítulos de secciones y similares, nunca para destacar bloques grandes de texto.
- El **subrayado** no es adecuado estéticamente y tiene un significado específico en la Web: señalar los hipervínculos.
- El texto en **colores** tiene este mismo problema, y en el caso de utilizarlo es necesario que los tipos contrasten con el color del fondo y que no se aproximen a los colores habituales de los enlaces, el azul y el rojo.

Recomendación

En general, se recomienda utilizar estos medios de enfatización con cuidado, pues destacar muchos elementos produce despiste y sensación de mareo en el usuario.

8.10. Sistemas de ayuda a la navegación

Para impedir el desbordamiento cognitivo, la navegación debe ser lo más intuitiva posible, sin abusar ni de los elementos multimedia ni de los enlaces innecesarios; se debe estructurar la información de manera ordenada y sencilla, utilizando herramientas de orientación o navegación.

Para no perder el rumbo de la lectura y encontrar nuevas rutas de navegación, las ayudas a la navegación han de estar claramente visibles para evitar la pérdida de orientación del usuario. Destacamos las siguientes:

8.10.1. Los sumarios, índices o tablas de contenido

Fuente: Web style guide online. http://webstyleguide.com/wsg3/index.html

8.10.2. Los mapas de contenido

Mapa global de contenido

Fuente: Universidad Politécnica de Valencia. http://www.upv.es/otros/mapa-web-es.html

8.10.3. La rama jerárquica

Rama jerárquica en la UPV

Fuente: Universidad Politécnica de Valencia.
http://www.upv.es/otros/mapa-web-es.html

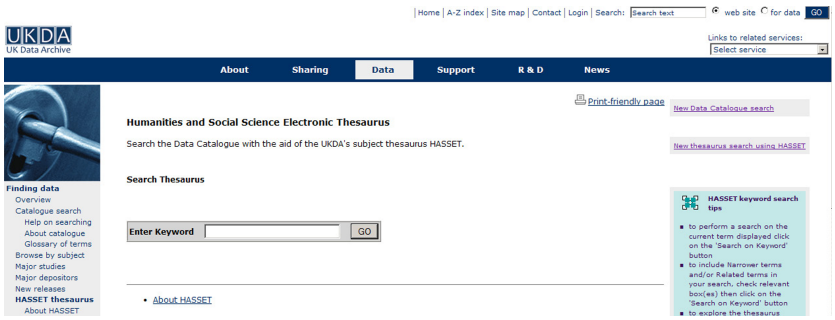
8.10.4. Los mapas táctiles

Mapa táctil

Fuente: Institut Cartogràfic de Catalunya. http://www.icc.es/web/content/es/index.html

8.10.5. La estructura de *frames* o subpantallas

Frames o subpantallas



Fuente: Hasset Humanities and Social Science Electronic Thesaurus. http://www.data-archive.ac.uk/search/hassetSearch.asp

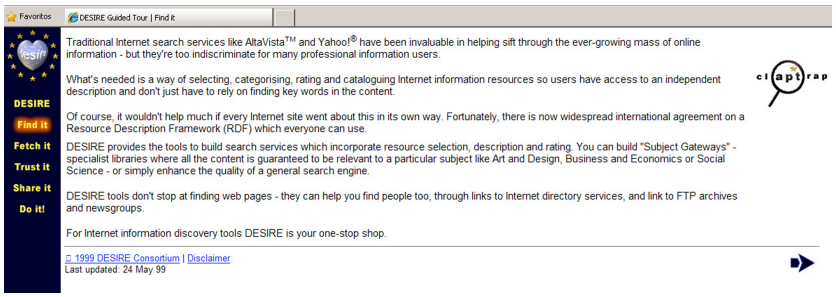
8.10.6. Otros sistemas de ayuda a la navegación

Además de las ayudas enumeradas, podemos destacar:

- Incluir un identificador de sitio en cada página que permita a los usuarios familiarizarse con la sede web como un todo. Habitualmente se utiliza un **logotipo** en la parte izquierda de la pantalla que da sensación de contexto.
- Facilitar el establecimiento de límites entre páginas. Cada página debe estar enlazada con la página principal y con la página de búsqueda (formulario del motor de búsqueda).
- No cambiar el color de los enlaces que existe por defecto; los enlaces que no se han visitado habitualmente son azules, y los que se han visitado, de color rojo. Esto permite en cualquier sede web saber qué zonas de ésta se han visitado ya; también ayuda a formar un modelo mental de la sede.
- Aquellas incluidas en el navegador, como el botón de retroceso, el historial de los nodos visitados, y el archivo de marcadores o *bookmark*.

8.10.7. Visitas guiadas

Pantalla de la visita guiada de la sede web de DESIRE



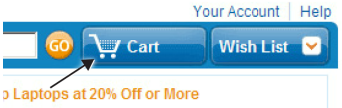
Fuente: DESIRE. Guided tour. http://www.desire.org/html/tour/house.html

8.10.8. Metáforas

Las **metáforas** pueden utilizarse como herramientas de navegación, ya que utilizan conceptos y modelos del mundo real con los que el usuario está familiarizado. Son representaciones de conceptos dirigidos a la comprensión del usuario del sistema y equivalen al tipo de información que se intenta transmitir.

Lectura recomendada

P. Diaz; N. Catenazzi; I. Aedo (1996). *De la multimedia a la hipermedia*. Madrid: Ra-ma.



Un ejemplo muy común de metáfora en la Web es la cesta de la compra.

8.11. Usabilidad y accesibilidad

8.11.1. Usabilidad

La **usabilidad** (*usability*), o **calidad de uso**, y sus técnicas han sido aplicadas dentro del área de estudio denominada **interacción persona-ordenador** (IPO) o *human-computer interaction* (HCI), centrada en la interacción entre usuarios y sistemas informáticos.

Entre las disciplinas con las que se relaciona podemos enumerar la psicología cognitiva y de la conducta, la ergonomía, la antropología, la sociología y las ciencias de la computación, el diseño centrado en el usuario o la evaluación de la calidad.

Según Nielsen, la usabilidad es un atributo que mide cómo de fácil es el uso de las interfaces de usuario.

Lectura recomendada

J. Nielsen (2003). *Usabilidad. Diseño de Sitios Web*. EE. UU.: Prentice Hall.

La norma **ISO 9241-11** la define de modo más general como la medida en que un producto puede ser utilizado por usuarios específicos para lograr objetivos con eficacia, eficiencia y satisfacción en un contexto de uso específico, es decir, como un método de diseño y evaluación.

El **contexto de uso** serían los usuarios, las tareas, el equipo (hardware, software y materiales) y los entornos físicos y sociales en los que se utiliza un producto, y el **usuario**, la interacción del individuo con el sistema.

La usabilidad no sólo puede ser definida como atributo de calidad de un producto, sino también como metodología de diseño y evaluación de éste. Es una disciplina en constante evolución e íntimamente ligada a las disciplinas que la utilizan desde un enfoque de diseño centrado en el usuario.

El **diseño centrado en el usuario** (DCU) o *user-centered design* (UCD) tiene un procedimiento de trabajo que, como su nombre indica, pone al usuario en el punto clave del diseño. En muchas ocasiones, el concepto de experiencia del usuario también se asocia con usabilidad. La **experiencia del usuario** es un concepto mucho más amplio que el de usabilidad; de hecho, podemos decir que la experiencia del usuario engloba a la usabilidad.

Lectura recomendada

Human-centered design processes for interactive systems (ISO 13407 Model, 1999).

Es uno de los principales recursos sobre usabilidad en la Web que mantiene una completa propuesta de métodos y técnicas de usabilidad en el proceso de diseño web. Todos los materiales son de dominio público y pueden ser libremente descargados. En inglés, **UsabilityNet**: http://www.usabilitynet.org

Revista electrónica *Open Access* y de carácter multidisciplinar que pretende servir de herramienta para la difusión, divulgación e intercambio de conocimiento entre desarrolladores e investigadores en el campo del diseño y la tecnología. **NSU (No sólo usabilidad)**: http://www.nosolousabilidad.com

J. Nielsen (2000). *Usabilidad: diseño de sitios web*. Madrid: Prentice Hall y useit.com: usable information technology. Disponible en: http://www.useit.com

P. J. Lynch; S. Horton; J. Escofet (2004). *Manual de estilo web: Principios de diseño básico para la creación de sitios web*. Barcelona: Editorial Gustavo Gili, S.L. ISBN: 8425219426.

Este manual básico para diseñadores de sitios web ofrece consejos claros y concisos sobre cómo crear páginas y sitios web prácticos y bien diseñados. Contiene información sobre arquitectura de la información, mantenimiento de un sitio web y el diseño multimedia. Se puede consultar la versión en línea en inglés, *The Web Style Guide*, en: http://webstyleguide.com/wsg3/index.html

L. Rosenfeld; P. Morville (2002). *Information Architecture for the World Wide Web*. Beijing: O'Reilly (2.ª ed.).

G. Kelly; E. Cotler (2002). *Rediseño de sitios web*. Madrid: Prentice Hall.

A. Knapp Bjerén (coord.) (2002). *La experiencia de usuario*. Madrid: Anaya Multimedia.

A. Floría Cortés (2000). *Recopilación de Métodos de Usabilidad* . SIDAR. Disponible en: http://www.sidar.org/recur/desdi/traduc/es/visitable/Herramientas.htm

H. Yusef; F. J. Martín Fernández; I. Ghzala (2004). "Diseño Web Centrado en el Usuario: Usabilidad y Arquitectura de la Información". *Hipertext.net* (núm. 2). ISSN: 1695-5498. Disponible en: http://www.hipertext.net/web/pag206.htm

8.11.2. Accesibilidad

Un concepto unido directamente al de usabilidad es el de **accesibilidad**, un concepto más amplio, ya que se refiere al máximo rango posible de usuarios, incluyendo a personas con discapacidad o limitaciones derivadas para el acceso, como el software y hardware empleados, ancho de banda de la conexión, etc.

La norma ISO/TC 16027 define accesibilidad como la facilidad de uso de modo eficiente, eficaz y satisfactoria de un producto, servicio, entorno o instrumento por personas que poseen diferentes capacidades.

Para poder comprender mejor la definición de accesibilidad dentro del contexto de la Web, debemos considerar los tipos de limitaciones del usuario que pueden impedir su acceso a la información:

- Dificultades visuales: como la ceguera, la visión reducida y los problemas en visualización de color.
- Dificultades auditivas.
- Dificultades motrices.
- Dificultades cognitivas y de lenguaje.

Ejemplo

Problemas del usuario para ver, escuchar, moverse o procesar algunos tipos de información, dificultades en la lectura o comprensión de un texto, no ser capaz de utilizar un teclado o un ratón, tener una pantalla que sólo muestre texto o que sea pequeña, o una conexión a Internet lenta, no comprender el idioma original del documento, situaciones en las que sus ojos, oídos o manos estén limitados (como en la conducción de un vehículo, trabajar en entornos ruidosos, etc.) y tener versiones anteriores o diferentes del navegador o sistema operativo.

Por lo tanto, la accesibilidad hace referencia a que los productos y servicios puedan ser utilizados por los usuarios con efectividad, eficiencia y satisfacción en un contexto de uso determinado.

"Un diseño web será accesible cuando sea usable por más personas en sus necesidades y preferencias".

H. Shawn Lawton (2002). "Understanding Web Accessibility". En: *Constructing Accessible Web Sites. Glasshaus* (abril). ISBN: 1904151000. Disponible en: http://www.macromedia.com/macromedia/accessibility/pub/acc_sites_chap01.pdf

Webs recomendadas

Normativa y legislación en materia de accesibilidad

Fundación Sidar-Acceso Universal Seminario SIDAR. Disponible en: http://www.sidar.org/mapa/index.php

Normas técnicas y pautas en materia de accesibilidad web

Web Accessibility Initiative (WAI). Disponible en: http://www.w3.org/WAI/. En español: http://www.w3c.es/Traducciones/es/WAI/intro/accessibility

En Internet se puede encontrar una gran variedad de herramientas que permiten comprobar la accesibilidad de un sitio web.

Directorios de herramientas para el análisis, la evaluación y la validación automáticas de la accesibilidad web

Web Accessibility Evaluation Tools. Disponible en: http://www.w3.org/WAI/ER/tools/

Herramientas de validación W3C. Disponible en: http://www.w3.org/Status, http://www.w3.org/QA/Tools/

Accesibilidad web: Lourdes González Perea y José Ángel Martínez Usero. Sedic. 2006. http://www.sedic.es/autoformacion/accesibilidad/presentacion.htm

Jens Meiert. UITest.com: Analysis, Validators, Checks, SEO Tools. http://uitest.com/en/analysis/

FLFSoft, Inc. HTML Validators. http://www.flfsoft.com/html/html_validators.html

P. Morville (2004). *User Experience Design. SemanticStudios* (21 de junio). Disponible en: http://semanticstudios.com/publications/semantics/000029.php

Discapnet, portal de la discapacidad. Disponible en: http://www.discapnet.es/

Formación

L. González Perea; J. Á. Martínez Usero (2006). *Accesibilidad web. Unidad de autoformación online*. Madrid: Sedic. Disponible en: http://www.sedic.es/autoformacion/accesibilidad/presentacion.htm