Técnicas de generación de contenido

Ana Belén Gil González [1] and Ana de Luís Reboredo [1]

[1] University of Salamanca, Plaza de los Caídos s/n – 37002 – Salamanca, Spain
{abg, adeluis }@usal.es

Resumen: HTML es una aplicación muy sencilla de las recomendaciones contenidas en el metalenguaje SGML. A lo largo de este capítulo se presentará cómo surge en la evolución de los lenguajes de marcado el HTML y estudiaremos su sintaxis para realizar sencillas páginas web. Además se mostrarán los diferentes formatos con los que se trabaja y con los que se gestionan las hojas de estilo o CSS. HTML, XHTML y las hojas de estilo en cascada (CSS). Estos elementos son muy importantes para la creación de cualquier página web.

Palabras clave: Web semantica

Abstract. HTML is a very simple application of the recommendations contained in the SGML metalanguage. Throughout this chapter it will be presented how HTML emerges in the evolution of markup languages and we will study its syntax to make simple web pages. It will also show the different formats with which you work and with which you manage style sheets or CSS. HTML, XHTML and Cascading Style Sheets (CSS). These elements are very important for the creation of any web page.

Keywords: Semantic Web

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

1. Introducción

La popularización e interés que ha despertado el lenguaje HTML en tan poco tiempo (menos de un lustro) es sorprendente. Ninguno de sus diseñadores lo hubiera sospechado en el momento de su creación. HTML es una aplicación muy sencilla de las recomendaciones contenidas en el metalenguaje SGML. A lo largo de este módulo estudiaremos cómo surge en la evolución de los lenguajes de marcado el HTML y estudiaremos su sintaxis para realizar sencillas páginas web a las que añadiremos la definición de formato con ayuda de las hojas de estilo o CSS. HTML, XHTML y las hojas de estilo en cascada (CSS) resultan vehículo imprescindible para la creación básica de cualquier página web [1-3].

1.1 Páginas Web Estáticas y Dinámicas

Una de las divisiones que podemos realizar entre todos los tipos de páginas Web existentes podría ser entre estáticas y dinámicas. Dedicaremos unas páginas a describir brevemente sus diferencias para centrarnos a lo largo del módulo en el desarrollo de páginas estáticas con la inclusión de hojas de estilo en cascada.
Por un lado las páginas Web estáticas, se presentan con ausencia total de movimiento y sin funcionalidades más allá de los enlaces y se construyen con HTML/XHTML. Son fáciles de construir pero muy difíciles de mantener. No poseen ningún tipo de interactividad con el usuario, pero constituyen la base fundamental e inicial de cualquier desarrollo.
Por otro lado están las páginas web dinámicas, dotadas de efectos especiales y con las que el usuario puede interactuar en procesos como consultas, accesos complejos y seleccionados por el usuario, etc. Son páginas más complejas y versátiles que las estáticas, por lo que para su realización son necesarios otros lenguajes de programación y/o tecnologías más allá del simple HTML/XHTML.
Tenemos dos tipos de páginas dinámicas teniendo en cuenta donde se ejecutan:

- Páginas dinámicas de Cliente: son aquellas que se procesan en el cliente, esto es se ejecutan en el navegador del usuario.
- Páginas dinámicas de Servidor: son aquellas que se ejecutan en el servidor.

Aunque las páginas dinámicas necesiten otros lenguajes aparte del HTML éste sigue siendo la base del desarrollo de cualquier página Web. Generalmente una página dinámica tiene el código de los otros lenguajes de programación embebido dentro del código HTML. Muchos son los ejemplos de este tipo de páginas tales como horóscopos, tiendas virtuales…

1.1.1 Páginas dinámicas de Cliente

Toda la carga de procesamiento, de los efectos y funcionalidades la soporta el navegador. Su principal desventaja es que dependen de las características del navegador tales como el tipo, las versiones, etc. Sin embargo son muchas las ventajas debido a que las páginas descargan al servidor de tareas, ofrecen respuestas inmediatas a las acciones del usuario y permiten el uso recursos de la máquina local. El código necesario para crear los efectos y funcionalidades se incluye dentro del mismo archivo HTML y es llamado SCRIPT [4-7].
El navegador interpreta los scripts de cliente y los ejecuta para realizar los efectos y funcionalidades. Para escribir páginas dinámicas de cliente existen varios lenguajes, algunos de ellos son: Javascript, Visual Basic Script (VBScript), Dynamic HTML (DHTML), Cascade Style Sheets

ISBN: 978-84-9012-862-6

Colección Aquilafuente: AQ-242

(CSS), Applets de Java. Las páginas dinámicas de cliente se escriben en dos lenguajes de programación principalmente Javascript y Visual Basic Script (VBScript) que con el uso de los CSS resulta una práctica generalizada en cualquier desarrollo actual [7-10].
El Javascript Es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez. Por un lado permite la generación de acciones de JavaScript para la creación efectos especiales sobre páginas web, tales como crear contenidos dinámicos y elementos de la página que tengan movimiento o cambien de color, etc. Por otro lado permite que se ejecuten las instrucciones como respuesta a las acciones del usuario, con lo que podemos dotar de interactividad cualquier página.
El VBScript es el lenguaje de scripts de Microsoft para la creación de páginas web y sólo es compatible con Internet Explorer. Está basado en Visual Basic, un popular lenguaje para crear aplicaciones Windows, en su versión reducida. El modo de funcionamiento de Visual Basic Script para construir efectos especiales en páginas web es muy similar al utilizado en Javascript. Los recursos a los que se puede acceder también son los mismos que con Javascript. No aconsejado en la Web que tiene como objetivo llegar a cualquier tipo de cliente/navegador, por tanto si se utiliza deberá ser para ciertos entornos limitados tales como una INTRANET, donde sea conocida la tecnología utilizada.
DHTML o HTML dinámico, no es en sí mismo un lenguaje de programación sino que es una nueva capacidad de los navegadores modernos, para tener un mayor control sobre la página. Cualquier página que responde a las actividades del usuario y realiza efectos y funcionalidades se puede englobar dentro del DHTML (efectos en el navegador para mostrar y ocultar elementos de la página, modificar su posición, dimensiones, color, etc.). El DHTML nos da más control sobre la página, gracias a que los navegadores actuales incluyen una estructura para visualizar en páginas web denominada capa. Las capas se pueden ocultar, mostrar, desplazar, etc. Sin embargo para realizar cualquier acción en la página, como modificar la apariencia de una capa, necesitamos Javascritp o VBScript. Dentro del concepto de DHTML se engloban también las Hojas de Estilo en Cascada o CSS (Cascade Style Sheets) ya que se descargan del lado del cliente o navegador que es quien interpreta los formatos y el estilo de los documentos para ser impresos, visualizados o leídos por un intérprete.
Los Applets de Java son una manera de incluir programas complejos en el ámbito de una página web. Se programan en Java (precompilados) y por tanto se benefician de la potencia de este lenguaje. La ventaja de utilizar applets consiste en que son mucho menos dependientes del navegador que los scripts en Javascript siendo además independientes del sistema operativo del ordenador donde se ejecutan. Como desventajas en relación con Javascript, los applets son más lentos de procesar y tienen espacio muy delimitado en la página donde se ejecutan ya que no se mezclan con todos los componentes de la página ni tienen acceso a ellos.

1.1.2 Páginas dinámicas del Servidor

Con este tipo de páginas dinámicas se puede hacer todo tipo de aplicaciones web tales como agendas, foros, estadísticas, chats, etc. Son útiles en trabajos que acceden a información centralizada, situada en una base de datos en el servidor. Ejemplos habituales de este tipo de páginas son las de los bancos, la prensa electrónica, el comercio electrónico.
Las páginas dinámicas del servidor se suelen escribir en el mismo archivo HTML, mezclado con el código HTML y siguiendo el siguiente proceso:

- El cliente solicita una página

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242

- El servidor ejecuta los scripts de esa página y genera una página resultado (solamente contiene código HTML)
- El resultado se envía al cliente.
- El cliente puede interpretar el resultado sin lugar a errores ni incompatibilidades, puesto que sólo contiene HTML

Para escribir páginas dinámicas de servidor existen varios lenguajes, algunos de ellos son: *Common Gateway Interface* (CGI) comúnmente escritos en Perl, *Active Server Pages* (ASP), *Hipertext Preprocesor* (PHP), *Java Server Pages* (JSP).
El CGI es el sistema más antiguo que existe para la programación de las páginas dinámicas de servidor. Los CGI se escriben habitualmente en el lenguaje Perl, aunque se pueden utilizar otros lenguajes: C, C++ o Visual Basic. Actualmente se encuentra un poco desfasado por la dificultad con la que se desarrollan los programas y la pesada carga que supone para el servidor que los ejecuta.
ASP (*Active Server Pages*) es la tecnología desarrollada por Microsoft para la creación de páginas dinámicas del servidor. ASP se escribe en la misma página web, utilizando el lenguaje Visual Basic Script o Jscript (Javascript de Microsoft). Se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la página ASP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores. Los servidores que emplean este lenguaje son los que funcionan con sistema Windows. ASP permite acceso a bases de datos, sistema de archivos del servidor y en general a todos los recursos del servidor. Existen componentes ActiveX para múltiples usos, envío de correo, gráficas dinámicas etc.
Otro lenguaje de programación del lado del servidor es PHP (*Hipertext Preprocesor*), gratuito y de código abierto e independiente de plataforma. Es rápido y compatible con las bases de datos más comunes, como MySQL, Oracle, Informix, y ODBC. Posee una gran librería de funciones (matemáticas, de red…). Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores.
Finalmente JSP (Java Server Pages), son páginas de Servidor Java. Es una tecnología orientada a crear páginas web con programación en Java. Con JSP se puede crear aplicaciones web multiplataforma (Java es multiplataforma). Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. El motor de las páginas JSP está basado en los servlets de Java (programas en Java destinados a ejecutarse en el servidor) pero JSP resulta mucho más sencillo aprender que los servlets.
Las ventajas de este tipo páginas web dinámicas del lado del servidor son que el cliente no puede ver los scripts, ya que se ejecutan y transforman en HTML antes de enviarlos. Además son independientes del navegador del usuario, ya que el código que reciben es HTML fácilmente interpretable. Como desventajas se puede señalar que será necesario un servidor más potente y con más capacidades que el necesario para las páginas de cliente.

ISBN: 978-84-9012-862-6

Colección Aquilafuente: AQ-242

1.2 Conceptos generales de lenguajes de marcado

En los años 60, Charles F. Goldfab lidera el intento de IBM de resolver los problemas asociados al tratamiento de documentos en diferentes plataformas a través del Lenguaje de Marcas Generalizado, GML (*Generalized Markup Language*). El objetivo de GML era describir los documentos de forma que el resultado no dependiese de una determinada plataforma ni de una aplicación específica. No se trataba de la apariencia del documento, sino de la estructura lógica del mismo tal como la subdivisión en títulos, capítulos, páginas, etc.

Las *marcas* son códigos que indican a un programa el formato que tiene que dar al contenido que delimitan. De esta manera darle un formato concreto a un texto (p.e. detallar un tipo de letra o un tamaño) consistirá en delimitarlo con la correspondiente marca que indique cómo debe ser mostrado. Lo mismo ocurre con todos las demás características de cualquier información digital. El principal problema era que cada aplicación y cada fabricante utilizaban sus propias marcas para describir los diferentes elementos, era una auténtica torre de Babel en el mundo informático. Conociendo este sistema de marcas y conociendo a la perfección el sistema de marcas de cada aplicación sería posible pasar información de un sistema a otro sin necesidad de perder la estructura del documento. IBM solventó el problema de la interopetatibilidad haciendo que las marcas fueran texto plano o código ASCII accesible desde cualquier sistema. Y la norma se denominó GML (*General Modeling Language*).
GML evolucionó a lo largo de los años y pasó a manos de la Organización Internacional para la Estandarización **(ISO)** y se convirtió en SGML (ISO 8879), *Standart Generalized Markup Language*. Esta norma es la que se aplica desde entonces a todos los lenguajes de marcas, cuyos ejemplos más conocidos son el HTML o el RTF.
Los lenguajes de marcas no son equivalentes a los lenguajes de programación aunque se definan igualmente con el término "lenguajes". Son sistemas complejos de descripción de información digital, normalmente documentos, que si se ajustan fielmente a SGML, se pueden controlar desde cualquier editor ASCII. Es importante tener en cuenta que SGML no almacena el diseño, sino la estructura lógica del documento.
El formato de las marcas más utilizadas se compone de textos descriptivos encerrados entre signos de "menor" (<) y "mayor" (>), siendo lo más usual que exista una marca de principio y otra de final. Básicamente existen tres usos para los lenguajes de marcas:

i) los que sirven principalmente para describir el contenido de la información

ii) los que sirven para definir su formato

iii) los que realizan tanto la descripción del contenido como el formato

Las aplicaciones de bases de datos son buenas referencias del primero de los usos. Las hojas de estilo en cascada (CSS) son ejemplos típicos del segundo tipo, y por supuesto como veremos en esta unidad, el HTML es la muestra más conocida del tercer modelo.

1.3 SGML

Como metodología de edición de textos, SGML introduce conceptos de trabajo muy avanzados. Los tres tres conceptos básicos que introduce son:

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

- *El concepto de lenguaje de marcado generalizado* como un metalenguaje que sirve para definir lenguajes concretos que pueden adaptarse a cada dominio mediante una gramática que describe formalmente un tipo específico de documento o DTD (*Document Type Definition*);
- *El concepto de marcado descriptivo*, frente marcado procedimental. El marcado descriptivo describe, mediante las marcas o etiquetas definidas en la DTD, la estructura lógica de la información. La idea clave es que las marcas no determinan el procesamiento del documento de manera fija, ya que dicho procesamiento se determina a partir de las necesidades concretas, y se beneficia de la estructura lógica del documento caracterizada a través de sus marcas;
- *El concepto de independencia de la plataforma*. Como los documentos SGML únicamente contienen texto, éstos pueden ser procesados en distintas plataformas, trascendiendo el uso de dichos documentos a los sistemas que los crearon y utilizaron originariamente.

El uso de lenguajes de marcado descriptivo, como los definidos mediante SGML, inicialmente restringido a la publicación de documentos electrónicos, puede generalizarse a la construcción de aplicaciones informáticas. En esta generalización, las aplicaciones se describen mediante documentos que, posteriormente, se marcan mediante un lenguaje de marcado apropiado. Las aplicaciones surgen al procesar de forma adecuada dichos documentos.
Lo más novedoso es la idea de reemplazar los códigos y las macros procedimentales por **códigos declarativos** que separan el contenido (el valor funcional de los elementos de un documento) de su formato. Así, frente a una etiqueta procedimental que señala un bloque de texto como cursiva o subrayado, sin indicar el motivo del cambio de tipografía, una etiqueta declarativa indica la condición del bloque, si se trata de una cita bibliográfica, de un extranjerismo, de un tecnicismo, o de un fragmento que se desea enfatizar.
Esta posibilidad de **separar el contenido del formato** permite jugar de manera muy flexible con la información. Su estructuración, actualización, selección, combinación o presentación podrán ofrecerse según más convenga en cada oportunidad.
Pero SGML, aunque conocido por la utilización generalizada de etiquetas descriptivas ('Tags'), no constituye un conjunto predefinido de etiquetas, ni una sintaxis de *etiquetado* ('markup').
SGML es en realidad un metalenguaje que permite el diseño y control de un conjunto de etiquetas y de una sintaxis a la medida del usuario o de la aplicación. Su adaptación a las necesidades del uso se concreta en la *declaración de tipo de documento* (DTD).
HTML sigue las reglas de una DTD. Dentro de las múltiples posibilidades de SGML, la DTD de HTML describe un conjunto concreto de etiquetas de hipertexto y una sintaxis para utilizarlas en Internet [11-13].
El objetivo perseguido era por tanto disponer de un sistema de creación y distribución de documentos, que solucionase la problemática de mantener y distribuir información desarrollada en diferentes aplicaciones de edición, en un entorno de usuarios que precisan acceder a documentación común. SGML tenía una imperfección decisiva y es que resultaba demasiado complicado.

ISBN: 978-84-9012-862-6

Colección Aquilafuente: AQ-242

En su desarrollo habían aunado esfuerzos administraciones y empresas que durante años guardaban documentos en SGML, sin embargo no habían pensado en el usuario medio. En consecuencia, la creación de herramientas software para procesar documentos SGML las hacía muy caras y no tuvieron gran aceptación. Sin embargo llegó el World Wide Web y todo cambió…

1.4 Perspectiva Histórica del HTML

Internet es una enorme red de comunicaciones que permite la interconexión de sistemas informáticos, independientemente de su tipo y situación. Internet resulta el paradigma de entorno en cuanto a los comentados problemas en relación a la heterogeneidad de los formatos. Sobre los ordenadores, y aprovechando los servicios de comunicaciones de la red, se ejecutan diversos tipos de aplicaciones, que permiten realizar intercambios muy sofisticados de información. Para simplificar las dificultades de acceso a información dentro de un entorno tan extenso y heterogéneo como Internet, se ideó un nuevo servicio de información. Integraba un atractivo formato de presentación de datos, junto con un sistema para estructurar y enlazar tipos muy variados de información. Su nombre, el World Wide Web, 'la telaraña mundial'.

Lo que la mayoría de la gente suele olvidar es que Internet existía ya en los comienzos de los años setenta[1]. Los primeros mensajes de correo electrónico se enviaron en 1971 en EE.UU., y además del e-mail se utilizaban otros servicios como Usenet (foros de discusión), descarga de archivos a través de FTP, búsqueda de archivos con Archie, etc. Sin embargo moverse por las especializadas y técnicas ramas de esta telaraña era un privilegio sólo destinado a militares, científicos y trabajadores técnicos de universidades. Era necesario tener grandes conocimientos de sistemas operativos como UNIX e interfaces de usuario basadas en comandos debido a que por supuesto no existían las interfaces gráficas. Así pues el usuario medio no estaba preparado técnicamente para el acceso a esta red [14].

En este contexto un físico del CERN (Suiza), Tim Berners-Lee, cansado de los problemas de formato en el intercambio de información a través de la red y de los problemas a la hora de generar documentos de trabajo relacionados entre sí comienza a interesarse por las posibilidades del hipertexto, una metodología de organización de la información que, partiendo de un conjunto de documentos, permite el acceso desde uno a otro mediante enlaces.

Berners-Lee propuso en 1990 la creación de un sistema de documentos de hipertexto basado en Internet que mejoraría el intercambio de cualquier información. Esta propuesta y su puesta en marcha le han convertido en el padre de WWW. El proyecto fue desarrollado en principio con la colaboración de otros científicos del CERN, desarrollando los estándares y las herramientas básicas de la WWW. Los hitos fundamentales se pueden resumir en:

- Para la redacción de los documentos de hipertexto se desarrolló el lenguaje HTML.
- Para la transmisión de los documentos a través de Internet se desarrolló el protocolo HTTP (*HyperText Transfer Protocol*)
- Para la localización de documentos en Internet se desarrollaron las URLs (*Uniform Resource Locator*)
- Para la visualización de los documentos HTML se desarrolló un tipo de herramienta de visualización denominada navegador o *browser*.

[1] N.B.: resulta habitual la confusión que equipara Internet con World Wide Web.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

Hasta el 1993, la WWW no pasó de ser un modesto conjunto de documentos sobre Física diseminados por servidores de todo Internet. Sin embargo es a partir del citado año que surgen los primeros ISPs (*Internet Service Providers*) ofreciendo el acceso a Internet por un coste periódico. La WWW comenzó su expansión, suscitando un gran interés entre empresas y personas que empezaban a sumergirse en Internet. Además es en ese mismo año, 1993 que aparece el primer navegador gráfico, el NCSA Mosaic, que gozó de gran popularidad debido a su simplicidad y a las nuevas posibilidades de acceso a la información. El crecimiento del tráfico de datos en la WWW durante el año 1993 es del 341%.
A partir de 1993, el número de usuarios particulares en Internet comienza a crecer. En 1995, Internet se desvincula completamente del gobierno y del ejército de los EEUU, lo que supuso la apertura total del público general. Es a partir de 1995 cuando las empresas, animadas por el desarrollo técnico de WWW, comienzan a descubrir sus grandes posibilidades y se dispara la actividad en relación con el sector.
El enorme avance de las tecnologías durante el siguiente lustro hace que a finales de los 90, la WWW se convierta en el mayor entorno multimedia y el mayor repositorio de datos existente hasta nuestros días.
El World Wide Web se puede considerar una especie de "interfaz gráfica" para Internet, de manera que hace posible un acceso intuitivo y visualmente atractivo que hace accesible Internet al público general.

1.5 HTML: un lenguaje definido a partir de SGML

El HTML (*Hypertext Markup Language*) ideado por Tim-Berners Lee, se basó en estándares ya existentes, como una combinación de ASCII (*American Standard Code for Information Interchange*) y de SGML que ya hemos visto en anteriores apartados. Se puede decir que ASCCII es el mínimo elemento presente en la transferencia de datos entre ordenadores. Cualquier procesador, por sencillo que sea, lee y almacena información en este formato de modo que permite la interoperabilidad y el intercambio de dicha información. Sin embargo con ASCII no es posible resaltar ningún formato estructural tal como los títulos, subtítulos o formal tal como negritas, cursivas, aplicación de color, etc. Aquí es donde Lee hace uso del SGML, definiendo las opciones de formato representadas como etiquetas de texto, que se incluyen entre los signos de menor y mayor (< >) y denominadas *Tags*.
HTML define un lenguaje de marcado o marcas (*Tags*) que permite la creación de páginas Web. Dicho lenguaje derivado del SGML permite generar la estructura global de documento (Títulos, subtítulos, tablas, etc.) así como su formato (tipo de letra, tamaño, color, etc.). De esta forma HTML se convierte en una versión muy simplificada y mucho más comprensible de SGML, siendo un lenguaje de descripción para documentos de hipertexto, es decir para documentos Web con hipervínculos, como veremos en las siguientes secciones [15].

1.6 Versiones de HTML

La acelerada expansión de la WWW desde 1995 y el enorme avance en las tecnologías de desarrollo web eran un motivo creciente de nuevos conflictos. La existencia de tecnologías diversas para la creación de sitios web produce una falta de estandarización preocupante a mitad de los 90. La parte más afectada por esta falta de estándares fue el lenguaje HTML.
En 1994, Tim Berners-Lee y otros pioneros de la WWW fundan el *World Wide Web Consortium*, W3C, cuyo objetivo principal era definir los estándares de la WWW, incluyendo el lenguaje HTML. Así, en noviembre de 1995 apareció el primer estándar HTML, llamado HTML 2.0, desarrollado bajo el auspicio del IETF (*Internet Engineering Task Force*, en español Grupo de Trabajo

ISBN: 978-84-9012-862-6

Colección Aquilafuente: AQ-242

en Ingeniería de Internet) y recogido en la RFC-1866[2]. La siguiente versión de la especificación sería ya HTML 3.2 (las anteriores como HTML 3.0 no fueron sino meros borradores), aparecida en enero de 1997. Finalmente, en diciembre de ese mismo año apareció la última versión, llamada HTML 4, que fue revisada en abril de 1998 y ligeramente modificada en diciembre de 1999, dando lugar a la versión HTML 4.01.
Los principales aportes de la versión 4 de la especificación fueron:

- Aparición de los mecanismos para asociar información de estilo.
- *Scripts*, marcos y objetos.
- Mejoras en los formularios.
- Mejoras en aspectos de internacionalización y accesibilidad.

Sin embargo, hay que tener en cuenta que el HTML es un lenguaje que fue creado originalmente para representar simples documentos de texto. Aunque la versión 4.0 es adecuada para integrar elementos multimedia y de programación, dicha versión ha heredado de su origen la falta de robustez y una sintaxis muy ambigua que en las primeras versiones no era un inconveniente pero que resulta en el momento de la versión 4.0 uno de los grandes fallos del estándar HTML. El W3C solventará estas deficiencias en la revisión del estándar HTML concibiendo el XHTML 1.0., que fue aprobado por el W3C a principios de 2000. En los últimos apartados de este documento realizaremos una revisión más detallada del XHTML, una mezcla de HTML y XML lo que hace más estricta la sintaxis de HTML [16-20].

2 Conceptos iniciales de HTML

2.1 Estructura básica de un documento HTML

Hemos visto cómo la idea que subyace detrás del HTML era la de incluir información adicional al texto que se enviaba a través de Internet para controlar el formato y el diseño del contenido. El archivo HTML será interpretado por un cliente Web, por ejemplo un navegador, que eliminará toda la información adicional de la vista y creará los efectos deseados para el contenido.
Para diferenciar entre los contenidos textuales de una página web y la información sobre su formato y diseño, ésta última se inluye mediante etiquetas. Es decir es un lenguaje de descripción y tratamiento de texto, que nos permite:

- Escribir un texto.
- Estructurar el texto y aplicarle formatos utilizando los comandos del lenguaje, que son las etiquetas.

Un documento HTML es un fichero de texto *normal* formado por una jerarquía de elementos marcados y que podemos editar utilizando cualquier editor de texto. Cada elemento puede contener, a su vez, otros elementos y texto. Un elemento está delimitado por etiquetas (*Tags*), que pueden ser de tres tipos:

[2] http://www.ietf.org/rfc/rfc1866.txt http://www.ietf.org/rfc/rfc1866.txt

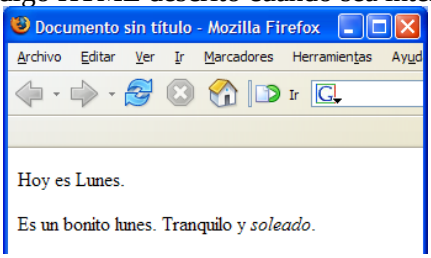
ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

- Etiquetas de inicio de elemento: contienen el nombre del elemento y sus atributos. Por ejemplo la etiqueta de inicio de párrafo con un atributo que le detalla que estará alineado al centro: `<p align="center">`
- Etiquetas de fin de elemento: contienen sólo el nombre del elemento. Por ejemplo: .
- Etiquetas de elemento vacío: útil cuando un elemento no tiene contenido. Por ejemplo la etiqueta de retorno de carro: <br />. Es equivalente a
</br>.

Muchos elementos, además de contenido, hemos visto como además pueden tener atributos. Un atributo está formado por un nombre y un valor. Los atributos de un elemento se especifican en su etiqueta de inicio (o de elemento vacío), indicando, en este orden, su nombre, el símbolo = y su valor, que suele incluirse entre comillas dobles o simples. Veremos múltiples ejemplos a lo largo de esta unidad.
He aquí un texto diseñado en HTML:

```
<p>Hoy es Lunes.</p>
<p>Es un bonito lunes.
 Tranquilo y <i>soleado</i>.</p>
```

El navegador interpretará este texto creando un nuevo párrafo, porque se ha utilizado la etiqueta de párrafo. A continuación mostrará el resto del texto hasta el cierre del párrafo y a continuación se creará un segundo párrafo mostrando la siguiente frase. La tercera frase que comienza con "Tranquilo…" no se mostrará en un párrafo nuevo porque no lo hemos especificado. La palabra "soleado" aparecerá en cursiva porque hemos utilizado la etiqueta de itálica antes de la palabra y la etiqueta de cierre después de la palabra. El resto del texto se mostrará normalmente y se cerrará el segundo párrafo. El contenido de un elemento es la porción del documento que se encuentra entre su etiqueta de inicio y su etiqueta de fin. El tipo de contenido de cada elemento está especificado, y debería ser respetado.
Así es como se mostrará el código HTML descrito cuando sea interpretado por un navegador:



Para incluir esta porción de texto en un documento HTML completo y que sea perfectamente visualizada por el navegador deberá incluir el formato obligatorio de un documento HTML:

inicio.html

```
<html>
<head>
</head>
<body>
        <p>Hoy es Lunes.</p>
        <p>Es un bonito lunes.
         Tranquilo y <i>soleado</i>.</p>
</body>
</html>
```

El principio y final de todos los archivos HTML tiene que abrirse y cerrarse con las etiquetas <html>. En un archivo HTML existen dos secciones principales, el encabezado (HEAD) y el cuerpo (BODY), expresados mediante sus etiquetas respectivas.

ISBN: 978-84-9012-862-6

Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

2.2 La cabecera del documento: <HEAD>

La cabecera contiene declaraciones globales para todo el documento (título, sistema de codificación de caracteres, hojas de estilo, meta-datos, etc.).

2.2.1 La Etiqueta <TITLE></ TITLE >

Contiene el títulos del documento. No tiene atributos. El navegador lo visualizará en su barra de título. Por ejemplo pruebe qué ocurre al visualizar la página anterior en el navegador si incluimos en la cabecera (HEAD) del código anterior el código:

```
<title> Primer Ejemplo</title>
```

2.2.2 La Etiqueta <META>

En la cabecera puede haber uno o varias etiquetas de este tipo. Éstas contienen información que puede resultarle útil al navegador o al servidor web. Brindan además información a los buscadores acerca de nuestra Web. Existen muchos tipos, algunos de ellos son:

- META KEYWORDS: definen la página mediante palabras clave, un ejemplo: `<meta name ="keywords" content="palabra1,palabra2,palabra3,palabra4">`
- META DESCRIPTION: Describe el contenido de nuestra página. `<meta name="description" content="Descripción de la web">`
- META LANGUAGE: Indica el idioma de nuestra página. `<meta name="language" content="spanish">`

Las etiqueta META, se utilizan además para generar cabeceras http, que es la información que se transmite junto con el documento mediante el protocolo http. Los nombres y funciones de las cabeceras se encuentran definidos en el estándar del protocolo http. Estos Tags tienen dos atributos HTTP-EQUIV y CONTENT. Así el siguiente ejemplo, está diciendo que el tipo de contenido del documento es textual para generación del HTML:

```
<meta http-equiv="Content-Type" content="text/html">
```

2.3 El cuerpo del documento: <BODY>

En esta etiqueta contenedora se sitúan todos los contenidos directamente representables del documento, es decir todo lo que se va a mostrar en la ventana principal del navegador. Tiene varios atributos que nos permiten modificar el aspecto de todo el documento HTML.

BGCOLOR = *color*. Este atributo permite modificar el color de fondo del documento. En HTML, un color puede definirse de dos maneras: utilizando su nombre en inglés para los dieciséis colores estándar establecidos (White, Black, red,…) o escribiendo su código hexadecimal[3] precedido del símbolo #.

En este momento ya habrá realizado su primer documento HTML, y por tanto su primera página Web. A lo largo del curso iremos aprendiendo el uso de las etiquetas y muchos más aspectos de la generación de documentos html.

TEXT= color. Este atributo nos permite modificar el color del texto de todo el documento.

Así podríamos definir los colores de fondo de la página y del texto de un documento HTML:

```
<body bgcolor="#009999" text="#990099">
…
</body>
```

NOTA SOBRE EL COLOR EN LA WEB:

[3] Paleta de colores de referencia rápida en http://www.htmlhelp.com/icon/hexchart.gif

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

Las unidades de color en Web siguen el modelo de color RGB. Se trata de una representación numérica de la forma de mezclar de manera aditiva los colores rojo, verde y azul (Red, Green y Blue) para producir un determinado color. Cada uno de los tres colores mezclados se representan de manera numérica en hexadecimal en función de su intensidad de 0% a 100%, el porcentaje se indica con un número entre 0 y N donde 0 significa 0% de intensidad y N significa 100%.
En Web se utiliza el modo 24bit donde N será 255 (FF), de esta manera el color: rojo: ff0000, verde: 00ff00, azul: 0000ff.

Especificación numérica	Descripción
#rrggbb	Especifica el valor RGB en forma hexadecimal con seis dígitos
#rgb	Reducido de la forma hexadecimal en la que se repiten los dígitos (como #dfd, que se expande a #ddffdd)
rgb(x,x,x)	Donde x es un entero entre 0 y 255 (rgb(0, 18, 0))
rgb(y%,y%,y%)	Donde y es un número entre 0 y 100 inclusive (rgb(0%, 80%, 0%))

Existen dieciséis colores estándar (VGA) para los que nos basta con escribir su nombre:

aqua (aguamarina)	**gray** (gris)	**navy** (azul marino)	**silver** (plateado)
black (negro)	**green** (verde)	**olive** (aceituna)	**teal** (cerceta)
blue (azul)	**lime** (lima)	**purple** (lila)	**white** (blanco)
Fucsia (fucsia)	**maroon (**marrón)	**red** (rojo)	**yellow** (amarillo)

Ejemplo recopilatorio.
Esto es todo lo que tiene que saber para iniciarse a realizar una página Web. Si utiliza el Bloc de notas como editor de textos, asegúrese de elegir GUARDAR COMO>TIPO>TODOS LOS ARCHIVOS en el cuadro de diálogo Guardar con el nombre iniciocolor.html; en caso contrario Windows añadirá la extensión .txt al archivo. Con un navegador ya puede abrir el archivo que ha creado.

_____ iniciocolor.html _____

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html" />
<title>Primer ejemplo</title>
</head>
<body bgcolor="#99CCCC" text="#000000">
<p>Hoy es Lunes.</p>
<p>Es un bonito lunes.
 Tranquilo y <i>soleado</i>.</p>
</body>
</html>
```

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242

3 Elementos básicos de HTML

3.1 El texto en HTML

El texto de un documento HTML se introduce como contenido de aquellos elementos que lo permitan. Los navegadores se encargan de dividir el texto en líneas dependiendo del ancho de la ventada de representación, tamaño de letra, etc.

3.1.1 Fuentes: la etiqueta <FONT>

La etiqueta FONT permite variar el tamaño, color y el tipo de letra de un texto determinado. Contiene atributos:

`SIZE = tamaño` Da al texto un determinado valor en puntos. Acepta valores entre 1 y 7 y también valores relativos (+/-) veces superior/inferior.

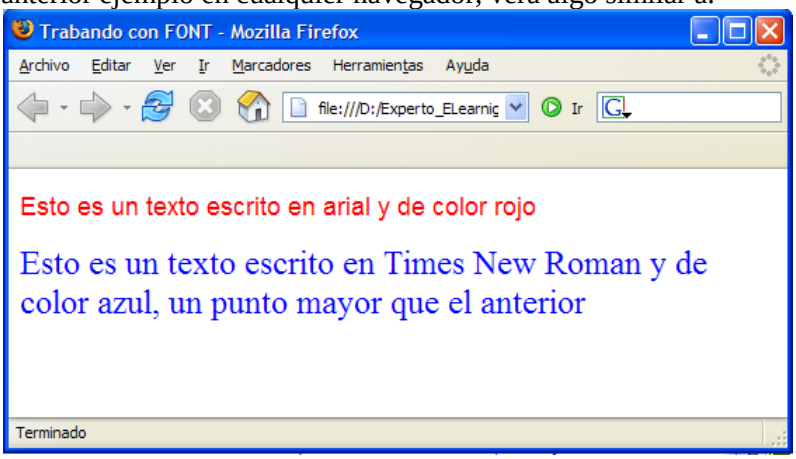
`COLOR = código de color` Escribe el texto en el código de color especificado, atendiendo la designacón del color en la web.

`FACE= nombre de fuente` Se indica qué fuente o lista de fuentes hay que usar. Es una buena idea incluir una lista de fuentes; si la primera no está presente en el ordenador del usuario, se utilizará la siguente de la lista. También es una práctica común escribir como última fuente un tipo genérico en lugar de una fuente particular.

fonttexto.html

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Trabando con FONT</title>
</head>
<body>
  <p><font size="4"  color="#FF0000" face="Arial, Helvetica, sans-serif">Esto es  un
  texto escrito en arial y de color rojo</font></p>
  <p><font size="+2"  color="blue" face="Times New Roman, Times, serif"> Esto es un texto
  escrito en Times New Roman y de color azul, un punto mayor que el anterior </font></p>
</body>
</html>
```

Si visualiza el anterior ejemplo en cualquier navegador, verá algo similar a:



3.1.2 Formato de texto

Estos tags, todos contenedores, aplican un formato al texto que contienen. Ninguno de ellos tiene atributos.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

<B> y <STRONG> La etiqueta <B> pone el texto en negrita y la <STRONG> indica que el texto es importante y generalmente lo pone en negrita también.
<I> y <EM> Ambas etiquetas ponen el texto en cursiva. La etiqueta <EM> indica además que el texto que contiene debe ser enfatizado.
<U> Subraya el texto que lo contiene. No se aconseja su utilización ya que el texto subrayado puede confundirse con los enlaces.
<SUP> Indica al navegador que el texto que contiene es un superíndice.
<SUB> Indica al navegador que el texto que contiene es un subíndice.

formatodeltexto.html

```
<html>
<head>
        <title> Formateando el texto </title>
</head>
<body>
        <p>Este es un ejemplo de formato de texto.</p>
        <p><strong>Este p&aacute;rrafo es importante.</strong></p>
        <p>Este texto no est&aacute; en negrita pero <i>yo estoy en cursiva </i></p>
</body>
</html>
```

En el ejemplo se puede observar que aparecen cosas ininteligibles como "párrafo". La razón es que existen caracteres que no pueden aparecer directamente en el documento, normalmente debido a que se trata de un carácter *reservado* (<, >, &), el editor no es capaz de representarlo, o el sistema de codificación del documento no lo permite (por ejemplo, en ASCII de 7 bits no es posible introducir tildes o la letra ñ). Para representar estos caracteres se hace mediante el uso de entidades, de modo que los Navegadores Web pueden identificarlos sin problemas. En la siguiente tabla se recoge la forma de referenciar a las entidades más habituales:

Carácter	Referencia
&	&amp;
<	<
>	>
á	á
é	é
í	í
ó	ó
ú	ú
ñ	ñ
?	¿
¿	¿

Para escribir en español es, sin embargo, más cómodo declarar el sistema de codificación de caracteres *ISO Latin 1* (ISO-8859-1), que permite utilizar todos los caracteres de los lenguajes de Europa occidental, sin necesidad de utilizar entidades. Para ello, se puede introducir en *head* el siguiente elemento:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

Para ampliar información sobre este respecto, tan útil en muchas ocasiones en la generación de documentos web, tenga en cuenta la Recomendación del W3C "HTML 4.01 *Specification*"[4] al respecto.

[4] http://www.w3.org/TR/1999/REC-html401-19991224/sgml/entities.html

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242

3.1.3 Comentarios

Los comentarios permiten incluir cualquier texto en el documento que los navegadores deben ignorar. Así, por ejemplo, el creador del documento puede dejar indicaciones útiles para la siguiente vez que tenga que modificarlo, ocultar temporalmente texto o elementos, etc. Se especifican entre "<!--" y "-->". Por ejemplo:

```
<!-- un comentario en una línea-->
<!-- comentario que ocupa más de una
línea-->
```

3.1.4 Formatos de Bloque

Las etiquetas de formato de bloque aplican un formato a un bloque entero de texto, en lugar de aplicarlo a palabras o frases como lo ya visto en el caso de las etiquetas de de formato de texto. Todos son etiquetas contenedoras.

<P> Indica el comienzo de un nuevo párrafo. Para lo cual introduce una línea en blanco cuando abre la etiqueta <p>. Tiene un único atributo, ALIGN, que indica la alineación del párrafo, que puede tomar valores: LEFFT (izquierda), CENTER (centrado), RIGHT (derecha).

<H_x> Por otra parte, es muy frecuente la necesidad de estructurar un documento en secciones (capítulos, apartados, etc.), y que cada una de éstas tenga un título. En HTML se pueden utilizar la etiquetas de título (en inglés *header*): <h1> </h1>, <h2> </h2,. . ., <h6> </h6>. Estos etiquetas son un tipo especial de párrafo, de tipo bloque. Cuanto menor es el número de la etiqueta de cabecera más importante resulta la sección y mayor es el tamaño del texto que etiqueta, como se puede ver en el siguiente ejemplo:

```
...
<body>
<h1>Cabecera 1</h1>
<h2>Cabecera 2</h2>
<h3>Cabecera 3</h3>
<p>
Un párrafo dentro de la H3
</p>
</body>
```

`<DIV>` Esta etiqueta se utiliza para separar un bloque de HTML en el documento, de manera que añade un retorno de carro antes y después del bloque. Tiene también el atributo ALIGN. En la actualidad es muy útil para trabajar con hojas de estilos (CSS) y capas.

3.1.5 Separadores

Los separadores nos permiten controlar el flujo del texto del documento insertando saltos de líneas, líneas horizontales, o impidiendo que se produzcan estos saltos de línea.

`<BR>` Si se desea forzar un cambio de línea, puede recurrirse al elemento *br* (etiqueta `<br/>`), pero sólo en situaciones justificadas. En la mayoría de las ocasiones, basta con dividir el texto en párrafos (es decir, englobar cada párrafo dentro de un elemento *p*).

`<HR>` Inserta una línea horizontal con un salto de línea antes y después de la línea, mediante la etiqueta. Tiene varios atributos, algunos de ellos:

- `SIZE = píxeles`: Indica el grosor de la línea en píxeles
- `WIDTH= píxeles/porcentaje`: Indica la anchura de la línea, mediante la medida absoluta (píxeles) o relativa (el porcentaje de la pantalla que debe ocupar la línea).
- `ALIGN=RIGHT/CENTER/LEFT`: Indica la alineación de la línea, centrada por defecto.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

3.2 Imágenes: el elemento IMG

La revisión 2.0 del estándar HTML incluyó una etiqueta básica para la inclusión de imágenes, la etiqueta en línea `<IMG>`. Esta etiqueta fue mejorando notablemente en revisiones posteriores debido a que resulta uno de los más importantes debido a la enorme inclusión de éstas en las páginas Web [21-25].

Los formatos de imagen principales que manejan la mayoría de los navegadores son GIF (*Graphic Interchange Format*), JPEG (*Joint Photographic Expert Group*) y PNG (*Portable Network Group*).

- El formato GIF utiliza un algoritmo de compresión sin pérdida de calidad y no soporta más de 256 colores. Permite transparencias, entrelazado y animación (denominándose GIFs animados). Resulta ser el formato ideal para pequeñas imágenes de pocos colores y con áreas uniforme del mismo color.
- El formato JPEG utiliza un algoritmo de compresión con pérdida de calidad pero soporta colores de 24 bit por píxel es decir hasta 16'7 millones de colores, lo que lo convierte en el formato ideal para imágenes fotográficas. Resuelve de manera excelente la definición de degradados pero no admite transparencias.
- El formato PNG, desarrollado conjuntamente por W3C y Compuserve para crear un nuevo estándar en transferencia de imágenes en Internet. Permite una compresión sin pérdida de calidad para imágenes con una gran cantidad de color, admite trasparencias. El que esté especialmente diseñado para Internet hace que incorpore un mecanismo que permite ver la imagen completa cuando sólo se ha cargado 1/64 del total del archivo.

El elemento en esta etiqueta <IMG> no puede tener contenido, y en él se especifica mediante atributos la siguiente información:

`SCR=dirección de la imagen` Indica dónde se encuentra la imagen que queremos insertar en el documento. Fuente de la imagen *src (source* en inglés): URL, absoluta o relativa, de la imagen (atributo obligatorio).

`ALT=texto` Texto alternativo (*alt*) descriptivo de la imagen, pensado para navegadores incapaces de representar imágenes y para navegadores que representen el documento mediante otros medios como sonido.

`ALIGN=left/right/center/top/botton/middle` Nos permite especificar cómo se alinea el texto alrededor de la imagen, no especifica la alineación de la imagen en el documento.

`BORDER =píxeles` Permite especificar el grosor en píxeles del borde que rodea a la imagen.

`HEIGT= píxeles y WIDTH= píxeles` Tamaño de la imagen determinado por su altura (HEIGT*) y* anchura *(*WIDTH). Ambos son atributos opcionales que permiten al navegador conocer el tamaño de las imágenes sin necesidad de descargarlas. Su utilización facilita a los navegadores el ir representando la página mientras descarga las imágenes, permitiendo así al usuario ver la página mientras se realiza la descarga. Por defecto el navegador muestra la imagen con sus dimensiones originales. Sin embargo podemos con ayuda de los atributos modificar las dimensiones de la imagen visualizada en el navegador.

A continuación se muestran dos ejemplos de utilización:

```
<img src="http://www.usal.es/Web -usal/imgs/s_r1_c2.gif " alt="Logotipo de usal" width="35" height="47" />
```

ISBN: 978-84-9012-862-6

Colección Aquilafuente: AQ-242

```
<img src=" imgs/s_r1_c2.gif " alt="Logotipo de usal" width="35" height="47" />
```

3.3 Vínculos: los elementos A

Una de las principales diferencias entre texto e hipertexto es que el hipertexto permite que el usuario salte de una sección o página de un documento a otra sección o página. Se puede establecer vínculos desde texto, imágenes completas o zonas de la imagen. Se puede crear vínculos a documentos, imágenes, archivos y software transferible, etc.
Los enlaces a otras páginas se pueden realizar mediante la etiqueta ancla (en inglés, *anchor*) en línea *a*. Su contenido es el texto (o imágenes) que el usuario puede pinchar para activar el enlace. El destino del enlace donde se quiere saltar se especifica mediante el atributo *href*. El siguiente ejemplo muestra un enlace a otra página dentro de un párrafo:

```
<p>
La <a href="http://www.usal.es">Universidad de Salamanca</a>, se encuentra en
Espa&ntilde;a.
</p>
```

Este texto estará subrayado y tendrá un color diferente del texto que lo rodea. Si el usuario pincha en el enlace irá a la página indicada por el campo *href* que en este caso es la de la Universidad de Salamanca. Al volver a la página original, las palabras "Universidad de Salamanca" habrán cambiado de color.
La dirección de destino puede ser codificada de forma absoluta o relativa. Una dirección relativa nos indica donde está el documento en relación al documento HTML de partida. Una dirección absoluta indica donde está el documento mediante una URL completa. También se puede apuntar a un fichero de texto, como se refleja en el siguiente ejemplo:

```
<a href="documento.doc"> Enlace aun fichero de texto</a>
```

Además, un vínculo puede apuntar también a diferentes puntos del documento HTML en el que se encuentra. Esto se realiza mediante las anclas, que son puntos del documento a los que nos interesa saltar. La etiqueta <A> también sirve para definir anclas, utilizando el atributo NAME, que indica el nombre del ancla.

```
<a name="ancla1"> Esto es un ancla</h1>
```

Para enlazar con el ancla, también utilizamos la etiqueta <A>, asignando al atributo HREF el nombre del ancla precedido del símbolo #.

```
<a href="#ancla1"> Esto es un enlace que apunta a ancla1</h1>
```

Si seguimos este enlace, el navegador se situará en el ancla referenciada. Esto es muy cómodo cuando se tienen documentos muy largo y se quiere saltar de una parte del documento a otra. Asimismo puede ayudarnos para construir un índice, con distintas secciones a las que se puede acceder directamente, dentro del documento HTML.
También podemos realizar vínculos que mandan un correo electrónico. Cuando el usuario hace clic en un vínculo de correo electrónico, desde un texto o una imagen, se abre una nueva ventana de mensaje en blanco (utilizando el programa de correo asociado al navegador del usuario). En la ventana de mensaje de correo electrónico, el cuadro de texto "Para" se rellena automáticamente con la dirección especificada en el vínculo del mensaje de correo electrónico ("mailto:correo@usal.es) incluso con el motivo del correo que queramos pasar como parámetro (?subject=asunto).
Ejemplo: `<a href="mailto:correo@usal.es"?subject=Contacto >correo</a>`

3.4 Listas: los elementos UL, OL y DL

Las listas permiten estructurar información en *puntos*, comenzando cada punto en una nueva línea. Existen tres elementos, todos de tipo bloque, para especificar listas, uno para cada uno de los tipos de lista permitidos: listas ordenadas, listas no ordenadas y listas de definiciones.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242

- Las listas ordenadas (elemento *ol*) asignan automáticamente un número a cada *ítem* de la lista. El contenido de este elemento es uno o más *ítems de lista* (elemento *li*).
- Las listas no ordenadas (elemento *ul*) son como las anteriores, pero no asignan números a los *ítems*.
- Las listas de definiciones (elemento *dl*) tienen como contenido elementos *dt* y *dd*. El primero permite especificar el término a definir, y el segundo, la propia definición.

A continuación se muestran ejemplos de listas, y cómo se verían en un navegador:

CÓDIGO HTML	VISTA EN EL NAVEGADOR

`<p> Lista no ordenada: </p>`
`<ul>`
`  <li>Cine.</li>`
`  <li>Televisi&oacute;n.</li>`
`  <li>Deportes.</li>`
`  <li>Lectura</li>`
`</ul>`
	Lista no ordenada:
	• Cine.
	• Televisión.
	• Deportes.
	• Lectura

`<p> Lista  ordenada: </p>`
`<ol>`
`  <li>Cine.</li>`
`  <li>Televisi&oacute;n.</li>`
`  <li>Deportes.</li>`
`  <li>Lectura</li>`
`</ol>`
	Lista ordenada:
	1. Cine.
	2. Televisión.
	3. Deportes.
	4. Lectura

3.5 Tablas: el elemento TABLE

Las tablas resultan muy útiles para estructurar la información en filas y en columnas. Los elementos fundamentales de una tabla HTML son las filas y las celdas (En lugar de filas y columnas). Esto es debido a que las tablas HTML se definen mediante etiquetas contenedoras y cualquier tabla se puede definir del siguiente modo:

- Una tabla contiene una o varias filas
- Cada fila contiene una o varias celdas
- Cada celda contiene elementos HTML válidos (imágenes, listas, etc.)

Existen bastantes elementos relacionados con tablas, pero aquí sólo se explican los más relevantes: *table (contenedor de la tabla)*, *th* (cabecera), *tr* (fila) y *td* (celda).
En HTML, una tabla se representa como un conjunto de filas. A su vez, cada fila contiene una o más celdas. En la misma tabla, todas las filas deben tener el mismo número de celdas, para poder representar las columnas correctamente alineadas.
Por defecto, cada celda se corresponde con una columna. El atributo *colspan* de *td* permite variar esta correspondencia, estableciendo su valor como el número de columnas por las que se debe expandir una celda.
HTML permite unir celdas, tanto verticalmente como horizontalmente. Esto se consigue con los atributos ROWSPAN y COLSPAN de la etiqueta <TD>. Antes de nada, conviene saber que los navegadores en general, no dibujan las tablas hasta que las han leído por completo. Cuando un

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242

navegador se encuentra con una etiqueta <TD> con atributo ROWSPAN o COLSPAN mayor que 1, el navegador tiene almacenado que esa celda abarca más de una fila o más de una columna (o ambas cosas) y predefine las celdas colindantes de esta celda.
A continuación se presenta un ejemplo donde se define una tabla con tres filas y cada una de ellas con tres celdas (o columnas). La primera de las filas está formada por celdas cabecera. La tercera de las filas colapsa sus dos últimas celdas, con su aspecto aproximado:

```
<table border="1">
<tr>
 <th>izda.</th>
 <th>centro</th>
 <th>drcha.</th>
</tr>
<tr>
 <td>1 izda.</td>
 <td>1 centro</td>
 <td>1 drcha.</td>
 </tr>
<tr>
 <td>3 izda.</td>
 <td colspan="2">3 drcha.</td>
</tr>
</table>
```

izda.	centro	drcha.
1 izda.	1 centro	1 drcha.
3 izda.	3 drcha.	

4 Elementos avanzados de HTML

4.1 Marcos: el elemento FRAME

Los *frames* o marcos permiten la división del espacio de visualización del navegador en áreas (los 'marcos') independientes. En cada una de esas áreas o subventanas se puede presentar un documento HTML diferente. La utilidad principal de los marcos es que nos permiten tener ciertos documentos HTML en pantalla siempre, mientras otros cambian [26-30].
El contenedor principal de los marcos es el <FRAMESET> o conjunto de marcos. Es un contenedor especial, ya que cuando queremos utilizar marcos, sustituye al contenedor <BODY>. El contenedor <FRAMESET> puede contener uno o varias etiquetas <FRAME>, que contienen la información de los marcos del documento. Es un archivo HTML que define el diseño y las propiedades de un grupo de marcos, donde se incluye el número de marcos, el tamaño, la ubicación de los marcos y el URL de la página que aparece inicialmente en cada marco. Así una página web que contiene dos marcos consta de tres documentos html.
A continuación se presenta un ejemplo donde definimos el conjunto de marcos (*frameset*), especificando que queremos dividir el documento en dos subventanas (como si fueran dos columnas), una que ocupe el 20% de la ventana y la otra del 80%. En cada una por supuesto hay un frame que hará la llamada al correspondiente documento HTML.

_____ frame.html_____

```
<html>
<head>
       <title>Ejemplo simple de marcos</title>
</head>
<frameset cols="20%, 80%">
       <frame name="frameIzq" src="pagcontenidoIzq.html">
       <frame name="frameDcho" src="pagcontenidodcho.html">
```


ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

```
<noframes>
        <p>Este documento contiene
        <ul>
        <li><a href="pagcontenidoIzq.html">Un documento</a>
        <li><a href="pagcontenidodcho.html.html">Otro documento</a>
        </ul>
</noframes>
</frameset>
</html>
```

Además los documentos HTML que estarán contenidos en los frames derecho e izquierdo. Que pueden ser los siguientes:

pagcontenidodcho.html

```
<html>
<head>
        <title>Frame derecho</title>
</head>
<body>
 <p> Frame derecho </p>
</body>
</html>
```

pagcontenidoIzq.html

```
<html>
<head>
        <title>Frame izquierdo</title>
</head>
<body>
 <p> Frame izquierdo </p>
</body>
</html>
```

Como se puede ver, tres son los principales elementos que intervienen en el proceso de creación de marcos:

- El contenedor *frameset* utilizado para determinar la disposición de los marcos en la ventana de visualización. Observar que sustituye al elemento *body* como raíz del cuerpo del documento. Para añadir versatilidad a los frames, los contenedores <FRAMESET> pueden anidarse.

  Contiene atributos ROWS y COLS, que nos permiten especificar la cantidad y el tamaño de las filas o columnas del conjunto de marcos. Para su especificación se escribe la lista con las achuras de las columnas o con las alturas de las filas, separadas por comas. Estas medidas pueden ser absolutas (píxeles) o relativas (porcentaje). Por ejemplo el código siguiente especifica que queremos tres columnas. La primera abarca el 20% de la ventana, la segunda 400 píxeles y la tercera el espacio restante: `<frameset cols="20%, 400, *">`
  También podemos indicar el grosor en píxeles de los bordes de los frames con el atributo BORDER=píxeles
- *frame* utilizado para definir los contenidos y apariencia de un marco concreto. Contiene distintos atributos.

  SRC= *url*, indica la URL del documento contenido en el marco.
  NAME=identificador, podemos asignar a cada marco un identificador para organización interna de la página y referenciar los marcos por su nombre en el conjunto de marcos.
- *noframes* utilizado para especificar el contenido alternativo a presentar por el navegador en caso de que éste no soporte el uso de marcos.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242

4.2 Formularios: el elemento FORM

Uno de los avances más significativos de la revisión 2.0 del estándar HTML fue la inclusión de formularios con todos sus elementos, tales como campos de entrada, *checkboxes*, *radio buttons*, *botones*, etc. Los formularios son elementos de bloque que pueden contener *controles*, mediante los cuales el usuario puede interactuar con la página, normalmente para enviar datos al servidor *Web* [31-35].

4.2.1 El Contenedor <FORM>

Todo formulario HTML está contenido en la etiqueta <FORM>. Los formularios pueden contener elementos HTML (tablas, listas, imágenes, etc.) junto con elementos propios de los formularios, que llamaremos controles. Esta etiqueta tiene tres atributos:

- ACTION=*url*, indica la URL a la que tenemos que enviar los datos que se introducen en el formulario. El formulario puede ser enviado a una dirección de correo electrónico o a un programa o script que procesa su contenido. Aunque puede ser cualquier URL válida, lo más usual es que se dirija a un fichero en el servidor que sea capaz de recibir e interpretar información tales como a una página activa (ASP, PHP, JSP,…) o a un CGI.
- METHOD=GET/POST, este atributo indica cómo se deben enviar los datos del formulario. Ambos métodos están relacionados con el protocolo http. Veremos más detalladamente este atributo en el apartado de envío de formularios.

ENCTYPE=*método de codificación*, es un atributo opcional que indica cómo se codifican los datos al enviarse. Su valor por defecto, `application/s-www-form-unlencoded`, nos permite enviar todo tipo de datos, excepto ficheros.

4.2.2 Elementos de entrada, la etiqueta <INPUT>

Esta etiqueta permite introducir en el documento HTML un campo de entrada de datos. A través de este campo de entrada el usuario podrá introducir datos. El valor del dato introducido en un campo es enviado como parámetro al servidor web. Los campos de entrada pueden ser de distinto tipo (campo de texto, radio botones, ect.), para indicar el tipo de campo se utiliza el atributo TYPE. Cada control tiene un nombre, especificado mediante el atributo *name*. Además, cada control tiene asociado un valor inicial (salvo excepciones, especificado en el atributo *value*) y un valor actual (introducido por el usuario o por un *script*).
Cuando el formulario es enviado, se envía el nombre de cada uno de sus controles, y su valor actual.

- Atributo TYPE=*Tipo de entrada*, nos permite especificar el tipo de elemento del formulario que queremos insertar en el documento HTML. Puede tomar valores TEXT, CHECKBOX, RADIO, HIDDEN, IMAGE, BUTTON, PASSWORD, SUBMIT, o RESET.
- Atributo NAME= *identificador*, permite establecer un identificador único para el campo de entrada. Este identificador nos permitirá referenciar el valor del campo de entrada. Se trata de un atributo obligatorio.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

- Atributo VALUE=*valor*, especifica en general un valor inicial para el campo de entrada. El atributo variará sus valores iniciales de acuerdo al tipo de elemento (TYPE) de entrada utilizado.

4.2.3 Tipos de elementos de entrada

En un formulario se pueden utilizar distintos tipos de elementos de entrada:

- *Botón <INPUT TYPE="button">*: puede ser de tres tipos: *submit* (envía el formulario), *reset* (establece el valor inicial en todos los controles del formulario) y *push* (utilizado para cualquier otra cosa, en combinación con *scripts*).
- *Checkbox <INPUT TYPE="checkbox">*: permite especificar valores *on/off*. Pueden utilizarse, en el mismo formulario, varios con el mismo nombre. Si se especifica el atributo *checked="checked"*, se inicializa a *on*.
- *Radio Botón <INPUT TYPE="radio">*: también permite especificar valores *on/off*, pero de tal forma que sólo uno de los que comparten el mismo nombre puede estar en *on* simultáneamente. Si se especifica el atributo *checked="checked"*, se inicializa a *on*.
- *Menú*: permite al usuario seleccionar una opción entre varias. Se crean mediante los elementos *select*, *optgroup* y *option*.
- *Campo de Texto (una línea) <INPUT TYPE="text">*: permite que el usuario introduzca una línea de texto. Tiene varios atributos propios: SIZE nos permite especificar el tamaño del campo en caracteres, MAXSIZE el tamaño máximo de los datos introducidos, y READONLY toma un valor booleano e indica si el campo puede ser sólo leído.
- *Campo de Texto (varias líneas) <TEXTAREA>*: se trata de una etiqueta contenedora que permite introducir un texto en un recuadro de más de una línea. Tiene un atributo NAME que identifica el nombre del parámetro que genera. Tiene otros atributos opcionales para especificar el número de columnas (COLS) y filas (ROWS), y READONLY que indica que el campo de entrada únicamente puede leerse y no escribirse.
- *Listas <SELECT> y <OPTION>*. Las listas desplegables nos permiten elegir uno o arios elementos de entre un conjunto de elementos. La etiqueta <SELECT> es el contenedor de la lista y tiene un atributo NAME que la identifica. Una etiqueta *select* puede contener una o varias etiquetas <OPTION>, que tendrán un atributo VALUE. Al enviar el formulario, el valor del parámetro correspondiente a la lista de la opción seleccionada es el que es remitido.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242

- *Fichero<INPUT TYPE="file">*: permite al usuario seleccionar un fichero del sistema local de ficheros. Mediante este control se pueden enviar ficheros al servidor.
- *Valor oculto<INPUT TYPE="hidden">*: no se muestra al usuario. Su valor será siempre el valor inicial, salvo que algún *script* lo modifique. Permite que los *scripts* puedan pasar valores al servidor y la aplicación que gestiona los datos del formulario.

4.2.4 Envío de un formulario

Cuando el usuario pincha en un botón de tipo *submit*, se produce el envío del formulario. Existen dos métodos de envío, que se especifican mediante el atributo *method* del elemento *form*:

- *GET*: utiliza el método GET de HTTP. Los parámetros se codifican concatenados en el URI especificado en el atributo *action* del elemento *form*. No se debe utilizar este método para operaciones no idempotentes.
- *POST*: utiliza el método POST de HTTP. Los parámetros se codifican en el cuerpo del mensaje HTTP.

Los parámetros a los cuales se hace mención son pares que contienen el nombre y valor actual correspondientes a todos los controles *con éxito* (en general, son aquellos que no se encuentren deshabilitados, y los botones radio y *checkboxes* que se encuentren en *on*).
Por otra parte, hay dos formas de codificar los parámetros a enviar al servidor, seleccionables mediante el atributo *enctype* del elemento *form*:

- *application/x-www-form-urlencoded*: se sustituyen los espacios en blanco por "+", y todos los valores no alfanuméricos y caracteres reservados por "%" seguido de su valor ASCII (dos dígitos en hexadecimal). Los nombres se separan de los valores por un carácter "=", y cada par nombre–valor se separa del resto por un carácter "&".

Por ejemplo: nombre=Juan+P%C3%A9rez&edad=29. Éste es tipo de codificación por defecto, y el único permitido con el método GET.

- *multipart/form-data*: más eficiente para el envío de grandes cantidades de datos, así como datos binarios. Debe ser utilizado, por ejemplo, para el envío de ficheros al servidor. El cuerpo del mensaje HTTP se separa en varias partes, cada una de las cuáles envía los datos asociados a un control.

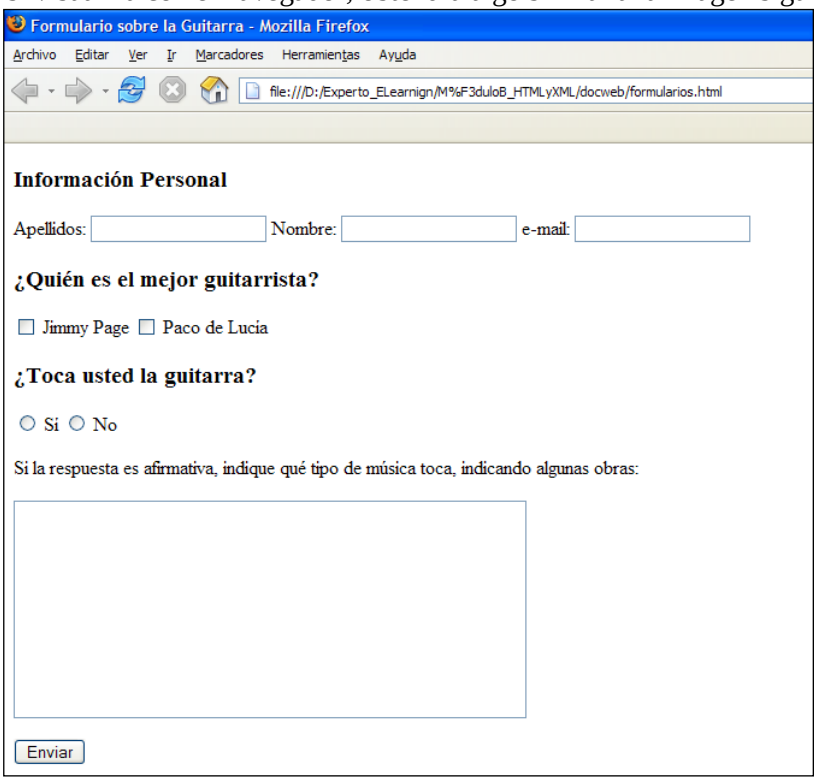
formulario.html

```
<form action="mailto:correo@dominio.es" method="post" encoding="multipart/form-data">
<h3>Información Personal</h3>
<p>
Apellidos: <INPUT name="apellidos" type="text" id="apellidos">
Nombre: <INPUT name="nombre" type="text" id="nombre">
e-mail:
<INPUT name="mail" type="text" id="mail">
</p>
<h3>&iquest;Qui&eacute;n es el mejor guitarrista? </h3>
<p>
<input name="selec_guitarrista"
type="checkbox" id="selec_guitarrista" value="Jimmy" />
Jimmy Page
```


ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

```
<input name="selec_guitarrista"
type="checkbox" id="selec_guitarrista" value="Luc&iacute;a" />
Paco de Luc&iacute;a
</p>
<h3>&iquest;Toca usted la guitarra? </h3>
<p>
  <input name="toca_guitarra" type="radio" value="Si">
  S&iacute;
  <input name="toca_guitarra" type="radio" value="No">
  No </p>
<p>
Si la respuesta es afirmativa, indique qu&eacute; tipo de m&uacute;sica toca, indicando
algunas obras:
</p>
<p>
<textarea name="informacion musical:" cols="50" rows="20" id="informacion musical:">
</textarea>
</p>
<p><input type="submit" value="Enviar" /></p>
</form>
```

De modo que si visualiza con el navegador, obtendrá algo similar a la imagen siguiente:



4.3 Validación de HTML

Es posible que en alguna ocasión cometa errores al codificar, es habitual. La diferencia entre el un desarrollador HTML con experiencia y uno sin ella no es simplemente la cantidad de errores que comete, sino sobre todo la eficiencia con la que resuelve cualquier error cometido. Lo mejor es utilizar un verificador de sintaxis. Existen fundamentalmente dos tipos de verificadores de sintaxis, online y offline. Dentro de las versiones en línea algunas le obligan a tener su página publicada.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

A continuación le propongo algunas direcciones donde puede encontrar recursos para realizar sus validaciones. Se debe de tener en cuenta que la mayoría de los editores de HTML incorporan este tipo de herramientas.
El validador de sintaxis del W3C (http://validator.w3.org) resulta de especial utilidad, pero encontrará muchos recursos para validar y optimizar su código HTML en otras direcciones como:

- SOFTONIC:

  http://www.softonic.com/seccion/532/Validadores_y_optimizadores_HTML

- http://www.htmlhelp.com/links/validators.htm

5 Del HTML al XHTML: Una introducción

XHTML es una redefinición de HTML 4 sobre XML. A pesar de detener distinto nombre, conserva la mayoría de los elementos y atributos de HTML. En apartados siguientes comentaremos en detalle las diferencias existentes entre HTML 4 y XHTML, pero adelantamos que las más destacables están en la ausencia de elementos y atributos relacionados con estilo (fuentes, colores, etc.), que ya en HTML 4 están desaconsejados, y en la existencia de unas normas más estrictas en la colocación de etiquetas de elementos.
En cuanto a los motivos por los que se originó la necesidad de redefinir HTML 4 como aplicación XML están fundamentalmente la necesidad de modularización de la especificación para adaptarse a los nuevos dispositivos así como la necesidad de establecer de la rigidez necesaria en la normativa para establecer una clara diferencia entre aspecto y contenido en la información.
En XHTML se refuerza la separación contenido/presentación, eliminando de la especificación aquellos elementos y atributos relacionados con el estilo. Esto tiene la ventaja de facilitar el cambio de la información de presentación para adaptarla a las características del dispositivo de salida concreto (asistente digital, teléfono móvil, ordenador, televisor, etc.)
Dado que los documentos XHTML son un tipo de documentos XML, todas las herramientas disponibles para trabajar con XML se pueden emplear también con XHTML.
En XHTML existen reglas estrictas acerca del formato que debe tener un documento.
Estás reglas se refieren por un lado a la buena formación del documento (establecen por ejemplo que todo elemento debe cerrarse adecuadamente) y por otro a la validez del mismo (indicando por ejemplo que dentro de un elemento <html> sólo puede haber un <body>). Todo documento XHTML, para ser tal, debe cumplir estas reglas, y el hecho de que las cumpla va a facilitar su procesado automatizado [36.
Modularización: a lo largo de este punto veremos que existen distintas versiones de XHTML, y que, a partir de la versión 1.1, se inicia un proceso de modularización de la especificación. Este proceso permitirá que los dispositivos de capacidades reducidas implementen únicamente ciertas partes de la especificación, y que los desarrolladores Web puedan adaptar mejor sus documentos a los dispositivos en los que se van a visualizar. Asimismo la modularización permitirá la extensión de XHTML sin rehacer de nuevo la especificación, bastará con añadir nuevos módulos aprovechando al máximo la capacidad de extensibilidad de XML.

5.1 Diferencias con HTML 4

Como ya indicamos, XHTML no es más que una redefinición de HTML 4 como aplicación XML. Como consecuencia de este hecho la semántica de los elementos y atributos de XHTML es exactamente la misma que en HTML 4. Existen sin embargo pequeñas diferencias entre los dos lenguajes de hipertexto, las principales de las cuales son:

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

- Los nombres de elementos y atributos deben escribirse en minúscula en los documentos XHTML, mientras que en HTML 4 era posible escribirlos en minúscula o mayúscula.
- Los valores de los atributos deben escribirse entre comillas (“ o ’) en XHTML. No son válidas construcciones del tipo `<font color=red>`.
- Todos los elementos tienen etiqueta de inicio o finalización (o elemento vacío). Por ejemplo: `<br/>`, `<p></p>`. No son válidas construcciones del tipo `<br>` o `<ul> <li>`A `<li>`B `</ul>`.
- La anidación de elementos debe ser correcta. No son válidas construcciones del tipo

Los elementos y atributos utilizados para especificar preferencias en cuanto al estilo de la información (como por ejemplo `<font>` o el atributo `bgcolor`) desaparecen.
Las características de presentación se establecen ahora mediante el uso de lenguajes específicos como CSS.
El atributo *id* sustituye al atributo *name* en: a, applet, frame, iframe, img y map.
Es obligatorio añadir al comienzo del documento XHTML una declaración *DOCTYPE* que referencia al DTD (Definición de Tipo de Documento) donde se indican las reglas de construcción del documento (como por ejemplo qué elementos y atributos son válidos y dónde puede aparecer cada uno). Ese DTD será específico de cada versión de XHTML [37].

5.2 Versiones de XHTML

Actualmente, existen varias versiones de XHTML:
XHTML 1.0 el más parecido a HTML 4, con tres variantes:

- **Transitional** permite el uso de las capacidades de presentación de HTML y está pensado para trabajar con navegadores con soporte de CSS limitado.
- **Strict** no se permite el uso de los elementos y atributos relacionados con aspectos de presentación. Pensado para ser usado con hojas de estilo CSS.
- **Frameset** permite el uso de marcos (*frames*) para dividir la ventana del navegador.

XHTML 1.1 nueva versión, que parte de XHTML 1.0 *strict*. No permite el uso de elementos y atributos relacionados con el estilo (pero sí, obviamente, hojas de estilo, como CSS). Se introduce el concepto de modularización.
XHTML Basic versión simplificada de XHTML 1.1 pensada para dispositivos de capacidad limitada de procesado, como televisores, teléfonos, móviles, PDAs, etc.
XHTML 2.0 se encuentra actualmente, y desde hace unos cuantos años, en proceso de estandarización (*Working Draft)* por parte del W3C (*World Wide Web Consortium*). Retoma la línea iniciada con XHTML 1.1 de modularizar XHTML, añadiendo nuevos módulos como *XML Events* y *XForms*. Estos módulos nacen con el objetivo de minimizar el uso de *scripts* dentro de documentos XHTML mediante la inclusión, como parte del propio lenguaje, de los medios necesarios para implementar las funcionalidades más importantes que requerían el uso de dichos *scripts* (eventos y formularios).
A continuación se muestra un ejemplo sencillo de documento XHTML 1.0 Transicional:

ISBN: 978-84-9012-862-6

Colección Aquilafuente: AQ-242

```
<!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=iso-8859-1" />
  <title>Documento sin t&iacute;tulo</title>
 </head>
  <body>
   Esta es una p&aacute;gina web en XHTML de transici&oacute;n.
  </body>
</html>
```

Con este pequeño manual damos por finalizado lo necesario en este módulo para crear sencillas páginas web. En los siguientes puntos indicamos referencias y direcciones de páginas web donde puedes descargar editores y herramientas que te facilitarán el trabajo así como cursos en línea donde profundizar en los conocimientos [39].

6 Hojas de Estilo en Cascada (CSS)

Las hojas de estilos en cascada o su acrónimo CSS (*Cascade StyleSheet)* son una especificación de reglas de formato que controlan el aspecto del contenido de una página Web. Cuando se utiliza CSS para formatear una página, se separa el contenido de la presentación de manera que el contenido de su página (el código HTML) reside en el archivo HTML. Las reglas CSS que definen la presentación de la información de la página, residen en otro archivo (una hoja de estilo externa) o en otra parte del documento HTML (normalmente en la sección <head>) [38].
Los estilos CSS aportan gran flexibilidad y control sobre el aspecto exacto que se busca en una página, desde la colocación precisa de elementos hasta el diseño de fuentes y estilos concretos.
El lenguaje de las Hojas de Estilo está definido mediante especificaciones CSS1 y CSS2 del *World Wide Web Consortium* (W3C). Son un estándar aceptado por toda la industria relacionada con la Web:

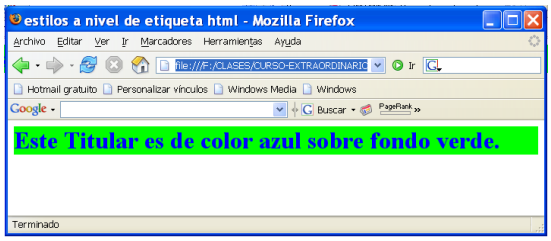
- CSS1: *Cascading Style Sheets, level 1. Se trata de una r*ecomendación W3C el 17 Diciembre de 1996, revisada en Enero de 1999. Está implementado correctamente en "todos" los navegadores actuales
- CSS2: *Cascading Style Sheets, level* 2. Recomendación W3C del 12 de Mayo de 1998. Cualquier CSS1 válido es CSS2 válido. Está implementado en los navegadores actuales.
- CSS3: en proceso (http://www.w3.org/Style/CSS/current-work…)

La mayoría de los lenguajes informáticos son estructurados porque cada elemento que contiene se procesa y se muestra en orden. CSS es tan amigable que no puede ser entendido como lenguaje de programación, sin embargo podemos decir que se trata de un lenguaje pero es declarativo y no estructural. En las hojas de estilo todo se expresa en reglas, que necesariamente no tienen por qué estar ordenadas. Cuando se solicita cualquier elemento formateado con un cierto estilo, el navegador busca la regla aplicable a dicho elemento. Estas reglas de estilo se asociarán a las etiquetas HTML básicamente de tres maneras:

1. Directamente a la etiqueta HTML. Es la forma menos recomendada porque no hace uso de las ventajas del CSS. Se define en la propiedad style los estilos a definir para dicha etiqueta.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

```
<html>
    <head>
    <title>estilos a nivel de etiqueta html</title>
    </head>
    <body>
        <h1 style="color:#0000ff;background-color:#00ff00">
          Este Titular es de color azul sobre fondo verde.
        </h1>
    </body>
</html>
```



2. Incrustadas en el propio documento HTML en el <HEAD> o cabecera de la página, tal y como muestra el siguiente ejemplo:

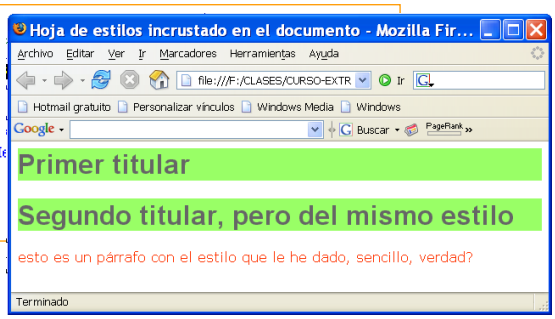
```
<html>
    <head>
        <title>Hoja de estilos incrustado en el documento</title>

        <style type="text/css">
            h1 {font-family: arial;
                color:#666666;
                background-color:#99FF66;
                }
            p   { font-family:Verdana, Arial, He
              color:#FF3300
                }
        </style>

    </head>

    <body>
        <h1>Primer titular</h1>
        <h1>Segundo titular, pero del mismo estilo</h1>
            <p>esto es un párrafo con el estilo que le he dado, sencillo, verdad?</p>
    </body>

</html>
```



Observamos cómo en la cabecera de la página <head> definimos las reglas de estilo dentro de las etiquetas <style>

- es recomendable especificar que estamos utilizando CSS en el campo TYPE de la etiqueta basado en texto
- También es importante indicar al navegador que lo que sigue se encuentra en un formato basado en texto
- En este ejemplo indicamos al navegador que en todos los lugares de esta página donde se utilice la marca h1 debe aplicar como estilo un color de texto determinado y color de

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

fondo el verde y que allí donde aparezca un párrafo, tendrá un tipo concreto de letra y color rojo….

3. Agrupando las reglas de estilo en un archivo independiente con extensión *.css. Es la manera óptima de trabajar con hojas de estilo. Se definen los estilos en un archivo separado con extensión css. (misestilos.css). Este mecanismo tiene múltiples ventajas: Optimizando recursos de desarrollo: Tiempo, peso de código, etc. De este modo podemos aplicar las mismas reglas de estilo a todas las páginas del sitio web. Resulta más ordenado tener lo referente a HTML en un archivo y las reglas de estilo en un archivo aparte. Se ahorra tiempo de transferencia. Cuando un navegador solicita una página, se le envía el archivo HTML y el archivo CSS, quedando guardado este último archivo en la caché de la máquina, así, en las sucesivas páginas que requieran el mismo archivo de estilos, ese mismo archivo se rescata de la caché y no requiere que el servidor web se lo reenvíe. la página HTML que tiene asociada una hoja de estilo en un archivo externo (misestilos.css). El archivo HTML es (paginaconestilo.html), para indicar el archivo de estilos externo, debemos agregar en la cabecera (head) del documento HTML la etiqueta señalada en la siguiente figura. La propiedad **href** hace referencia al archivo externo que afectará la presentación de esta página. En la propiedad **type**, indica al navegador cuál es el formato de archivo de texto para css. El atributo **rel** se usa para definir la relación entre el archivo enlazado y el documento HTML, en este caso de hojas de estilo.

```
<html>
<head>
<title>CSS en un archivo externo y vinculado</title>
<link rel="StyleSheet" href="misestilos.css" type="text/css">
</head>
<body>
<h1>Definiendo una hojas de estilo en un archivo externo</h1>
<p>
la hoja de estilo estará en un archivo separado que deberá tener la extensión
css.
</p>
</body>
</html>
```

Otro lugar donde se puede guardar sus hojas de estilo es en el ordenador del usuario. Esto tiene mucha relación con aspectos de la accesibilidad a la información web.

6.1 ¿Qué es una Regla CSS?

Una regla de formato CSS consta de dos partes:

- ★ el **selector**: es un término (como P, H1, el nombre de una clase o un ID) que identifica el elemento formateado (H1)

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

- la **declaración**: define los elementos de estilo. Siempre termina con punto y coma y luego todo se engloba entre llaves. Consta de dos partes:
  - La **propiedad** (font-family)
  - El **valor** de esta (arial)

Ejemplo de **regla** CSS: H1 {font-family: arial;}
Resulta sencillo el mecanismo, cada vez que utilicemos etiquetas H!, el texto contenido entre éstas será fuente arial.

6.2 Propiedades Básicas de CSS

Las propiedades que permiten definir el estilo o formato del documento están divididas en 5 grandes grupos:

1. Propiedades de fuentes
2. Propiedades de color y fondo
3. Propiedades de texto
    1. espaciado de palabras
    2. alineación
4. Propiedades de caja
    1. Margen
    2. Borde
    3. Relleno
5. Propiedades de clasificación
    1. Visualización
    2. listas

6.3 Reglas esenciales: Herencia y Cascada

Dos reglas esenciales Rigen los CSS: herencia y cascada
La Herencia de propiedades de estilo implica que las etiquetas de un documento HTML están organizadas de manera que unas engloban a otras. Por ejemplo todas las etiquetas se encuentran dentro del TAG <BODY> y otras como <EM> la podemos encontrar entre <P>, <H1>, <UL>, etc. Esta organización permite una relación padre-hijo de manera que los estilos definidos para etiquetas padres serán heredados por los hijos. Si hay existen declaraciones dentro de una regla a distintos niveles de cascada, la última heredará propiedades de la otra que no entren en conflicto.

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242

Se debe de tener en cuenta que no todas las propiedades se heredan y que es conveniente tener las especificaciones a mano para ver cuales se heredan y cuáles no.
La cascada. Los estilos definidos se encuentran en un fichero externo .css al que luego hacemos referencia a través de la etiqueta <LINK> o @IMPORT. A través de un bloque de estilos definidos en la etiqueta <STYLE> o directamente sobre la etiqueta HTML. La información se procesa en cada punto en dirección descendente con respecto a la ruta enumerada de búsqueda de las reglas. Cuando las reglas se solapan, la última regla que se lee tiene preferencia, de acuerdo a la jerarquía anterior.

6.4 Definición de estilos mediante clases

- En muchas situaciones una regla de estilo puede ser igual para un conjunto de marcas HTML, en esos casos conviene plantear una regla de estilo con un nombre genérico que posteriormente se puede aplicar a varias marcas de HTML. Sintaxis:

    1. La regla: la inicializamos con el carácter punto y seguidamente el nombre de la clase. (El nombre de la clase no puede comenzar con un número). El resto es lo mismo.

```
/* defino la clase alerta en mihoja.css */
.alerta{
  color:#FF0000;
  background-color:#FFFF00;
  font-style:italic;
}
```

    2. Luego, para indicar que una etiqueta queda afectada por esta regla agregamos la propiedad `class` y le asignamos el nombre de la clase (sin el punto).

```
<h1 class="alerta">T&iacute;tulo de nivel 1 en estilo de clase &quot;alerta&quot; </h1>
```

6.5 Definición de estilos mediante ID

Los identificadores (ID) funcionan de forma similar a las clases pero están limitados a su utilización con un sólo elemento. Los ID se definen utilizando el signo "#" seguido por el nombre del identificador y luego las propiedades del estilo:#NombreID {color: yellow}
Aplicándola luego a la etiqueta HTML mediante el atributo ID, por ejemplo:
Este parrafo ira en azul
La principal diferencia entre un ID o una clase para definir un estilo es que mediante un ID estamos identificando algún elemento de la página de forma univoca y por tanto sólo lo podemos utilizarlo con ese elemento. Esto se suele hacer porque luego posiblemente querremos realizar alguna acción sobre ese elemento (ej. para DHTML)

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

6.6 Modelo de Caja

Las cajas o cuadros permiten dominar el formato de una página web como elementos contenedores que se colocan en un determinado lugar de la página utilizando el atributo position. Los desarrolladores web tienden a organizar el contenido en orden, como si construyeran bloques. Así todo elemento que se crea dentro de una página HTML genera una caja. Por ejemplo el elemento o caja principal es el formado por las etiquetas <body> </body>, que a su vez puede contener cajas creadas por el resto de etiquetas (p, h1, h2..., id, ul, div, etc). De este modo además no es necesario utilizar la ventana del navegador y los elementos se pueden colocar en relación con cualquier objeto que los contenga.

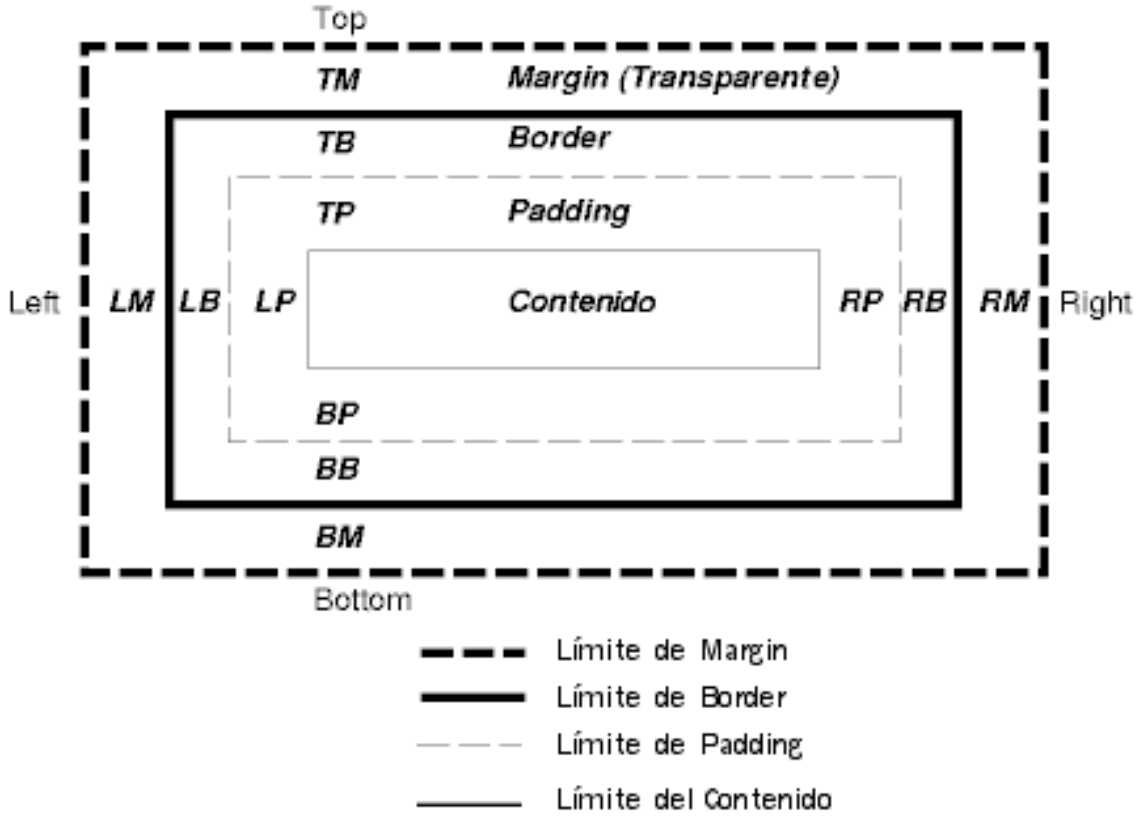
6.6.1 Propiedades

Los navegadores visualizan los documentos HTML en su ventana principal, tratando cada elemento que forma el documento como una caja o bloque, que deben posicionar siguiendo el orden y reglas definidas mediante reglas CSS. CSS proporciona herramientas conocidas como propiedades de caja, véase la imagen siguiente, para determinar su control:

- ⋆ Propiedad de margin: establece el espacio entorno al elemento
- ⋆ Propiedad de padding: margen interior establece el espacio entre el borde y el contenido del elemento
- ⋆ Propiedad border: determina el aspecto del borde que rodea al elemento

Se trata de propiedades compuestas, por ejemplo: border: 1mm solid #f33
Las tres áreas se dividen en cuatro segmentos que pueden tener valor diferente en cada lado de la caja: **top** (*superior*), **bottom** (*inferior*), **left** (*izquierdo*), **right** (*derecho*).

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242



6.7 Las capas

Para controlar la superposición de elementos en la página web se hace uso de la tercera coordenada, entendiendo que la página web es un elemento plano, esto es en dos dimensiones, trabajar con capas agrega esta tercera dimensión, la propiedad z-index. Esta propiedad: Acepta números enteros, cualquier elemento con un z-index de valor superior que su contenedor se colocará encima, si el valor de z-index<0 se colocará debajo del elemento que lo contiene.

6.8 Elementos organizativos

Las listas en HTML pueden generarse a través de: <OL></OL> etiquetas de lista ordenada, <UL>>/UL> lista de párrafos sin ordenar y <LI> cada elemento de la lista.
Haciendo uso de CSS las listas pueden ser mucho más flexibles y especificar cada uno de sus elementos. CSS controla el estilo de los elementos de lista utilizando las propiedades

- ⋆ **list-style-type**:
  - ➢ Listas sin ordenar: propiedades: disc, circle, square
  - ➢ Listas ordenadas:
    - – Decimal 1, 2, 3, …
    - – Decimal-leading-zero 01, 02, 03,…

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

- Lower-roman i, ii, iii,…
- Upper-roman I, II, III,…
- Lower-alpha a, b, c ,…
- Upper-alpha A, B, C, …

★ **list-style-image**: permite incluir su propia viñeta al principio de cada elemento de la lista

★ **list-style-position**: ubicación de la imagen

➢ Outside. A la izquierda del contenido

➢ Inside. Dentro del área que contiene los elementos de la lista

6.9 Pseudoclases y Pseudoelementos

Clasifican a los elementos basándose en características más allá de su nombre, atributos o contenido. La forma de definir un estilo para una pseudoclase es la siguiente:
etiqueta:pseudoclase {propiedad1:valor;..;propiedadN:valor}
Algunos ejemplos se describen a continuación.

6.9.1 Pseudoclases para los vínculos <A>:

★ :link - Nos dice el estilo de un enlace que no ha sido visitado.

➢ a:link { color: red; }

★ :visited - Nos dice el estilo de un enlace que ha sido visitado.

➢ a:visited { color: blue; }

6.9.2 Pseudoclases dinámicas

Cuando un elemento puede adquirir o perder una pseudo-clase a medida que el usuario interactúa con el documento.

★ :active - Nos dice el estilo de un enlace que está siendo pulsado.

➢ a:active { color: lime; }

★ :hover - Nos dice el estilo de un enlace sobre el que está pasando el ratón.

➢ a:hover {color: red; }

★ :focus - se aplica mientras un elemento tiene el foco (acepta eventos del teclado u otras formas de entrada de texto).

ISBN: 978-84-9012-862-6

Colección Aquilafuente: AQ-242

- a:focus { background: yellow }

6.10 Tipos de Medios

Las hojas de estilo permiten especificar cómo presentar un documento en diferentes medios: en la pantalla (screen), en papel (print), Dispositivo de mano (handheld), con un sintetizador de voz (aural), con un dispositivo braille (braille), etc.
Para su realización hay varios mecanismos. Bien seespecifica el medio de destino desde una hoja de estilo con las reglas-arroba: @media o @import (Ejemplo: @import url("loudvoice.css") aural; @media print { /* la hoja de estilo para impresión va aquí */ }). O bien especificando el medio de destino dentro del lenguaje del documento. Por ejemplo, en HTML 4.0, el atributo "media" en el elemento LINK especifica el medio de destino de una hoja de estilo externa: <LINK rel="stylesheet" type="text/css" media="print, handheld" href="foo.css">

7 Referencias y páginas Web de Interés

- Recomendaciones de la W3C: *http://www.w3.org*
- "HTML 4.01 Specification". *http://www.w3.org/TR/html4*
- "XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)". *http://www.w3.org/TR/xhtml1*
- "XHTML 1.1 - Module-based XHTML". *http://www.w3.org/TR/xhtml11*
- "XHTML Basic". *http://www.w3.org/TR/xhtml-basic*
- "XHTML 2.0 (Working Draft, July 26 2006)". *http://www.w3.org/TR/xhtml2*
- "Cascading Style Sheets, level 1". *http://www.w3.org/TR/REC-CSS1*
- "Cascading Style Sheets, level 2. CSS2 Specification". *http://www.w3.org/TR/REC-CSS2*
- ***http://www.w3.org/TR/REC-CSS1*** *- Cascading Style Sheets, Nivel 1 W3C Recommendation 17 Dec 1996, revised 11 Jan 1999*
- ***http://www.sidar.org/recur/desdi/traduc/es/css/cover.html - CSS2 en español***
- *http://jigsaw.w3.org/css-validator/ - Servicio de validación de CSS de W3C*
- *http://meyerweb.com/eric/css/ - Eric A. Mayer CSS*

http://www.csszengarden.com/ - Zen Gardens

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

References

1. Adrián Sánchez-Carmona, Sergi Robles, Carlos Borrego (2015). Improving Podcast Distribution on Gwanda using PrivHab: a Multiagent Secure Georouting Protocol. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 4, n. 1
2. Buciarelli, E., Silvestri, M., & González, S. R. (2016). Decision Economics, In Commemoration of the Birth Centennial of Herbert A. Simon 1916-2016 (Nobel Prize in Economics 1978): Distributed Computing and Artificial Intelligence, 13th International Conference. Advances in Intelligent Systems and Computing (Vol. 475). Springer.
3. Carlos Carvalhal, Sérgio Deusdado, Leonel Deusdado (2013). Crawling PubMed with web agents for literature search and alerting services. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 2, n. 1
4. Carolina González, Juan Carlos Burguillo, Martín Llamas, Rosalía Laza (2013). Designing Intelligent Tutoring Systems: A Personalization Strategy using Case-Based Reasoning and Multi-Agent Systems. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 2, n. 1
5. Casado-Vara, R., & Corchado, J. (2019). Distributed e-health wide-world accounting ledger via blockchain. Journal of Intelligent & Fuzzy Systems, 36(3), 2381-2386.
6. Casado-Vara, R., Chamoso, P., De la Prieta, F., Prieto J., & Corchado J.M. (2019). Non-linear adaptive closed-loop control system for improved efficiency in IoT-blockchain management. Information Fusion.
7. Casado-Vara, R., Novais, P., Gil, A. B., Prieto, J., & Corchado, J. M. (2019). Distributed continuous-time fault estimation control for multiple devices in IoT networks. IEEE Access.
8. Casado-Vara, R., Vale, Z., Prieto, J., & Corchado, J. (2018). Fault-tolerant temperature control algorithm for IoT networks in smart buildings. Energies, 11(12), 3430.
9. Casado-Vara, R., Prieto-Castrillo, F., & Corchado, J. M. (2018). A game theory approach for cooperative control to improve data quality and false data detection in WSN. International Journal of Robust and Nonlinear Control, 28(16), 5087-5102.
10. Chamoso, P., González-Briones, A., Rivas, A., De La Prieta, F., & Corchado J.M. (2019). Social computing in currency exchange. Knowledge and Information Systems.
11. Chamoso, P., González-Briones, A., Rivas, A., De La Prieta, F., & Corchado, J. M. (2019). Social computing in currency exchange. Knowledge and Information Systems, 1-21.
12. Chamoso, P., González-Briones, A., Rodríguez, S., & Corchado, J. M. (2018). Tendencies of technologies and platforms in smart cities: A state-of-the-art review. Wireless Communications and Mobile Computing, 2018.
13. Chamoso, P., Raveane, W., Parra, V., & González, A. (2014). Uavs Applied to the Counting and Monitoring Of Animals. In Advances in Intelligent Systems and Computing (Vol. 291, pp. 71–80). https://doi.org/10.1007/978-3-319-07596-9_8
14. Chamoso, P., Rivas, A., Martín-Limorti, J. J., & Rodríguez, S. (2018). A Hash Based Image Matching Algorithm for Social Networks. In Advances in Intelligent Systems and Computing (Vol. 619, pp. 183–190). https://doi.org/10.1007/978-3-319-61578-3_18
15. Chamoso, P., Rodríguez, S., de la Prieta, F., & Bajo, J. (2018). Classification of retinal vessels using a collaborative agent-based architecture. AI Communications, (Preprint), 1-18.
16. Corchado, J. M., & Aiken, J. (2002). Hybrid artificial intelligence methods in oceanographic forecast models. Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews, 32(4), 307–313. https://doi.org/10.1109/tsmcc.2002.806072
17. Corchado, J. M., & Fyfe, C. (1999). Unsupervised neural method for temperature forecasting. Artificial Intelligence in Engineering, 13(4), 351–357. https://doi.org/10.1016/S0954-1810(99)00007-2
18. Corchado, J., Fyfe, C., & Lees, B. (1998). Unsupervised learning for financial forecasting. In Proceedings of the IEEE/IAFE/INFORMS 1998 Conference on Computational Intelligence for Financial Engineering (CIFEr) (Cat. No.98TH8367) (pp. 259–263). https://doi.org/10.1109/CIFER.1998.690316
19. David Griol, Jose Manuel Molina, Araceli Sanchís De Miguel (2014). Developing multimodal conversational agents for an enhanced e-learning experience. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 3, n. 1
20. Fdez-Riverola, F., & Corchado, J. M. (2003). CBR based system for forecasting red tides. Knowledge-Based Systems, 16(5–6 SPEC.), 321–328. https://doi.org/10.1016/S0950-7051(03)00034-0
21. Fyfe, C., & Corchado, J. (2002). A comparison of Kernel methods for instantiating case based reasoning systems. Advanced Engineering Informatics, 16(3), 165–178. https://doi.org/10.1016/S1474-0346(02)00008-3
22. Fyfe, C., & Corchado, J. M. (2001). Automating the construction of CBR systems using kernel methods. International Journal of Intelligent Systems, 16(4), 571–586. https://doi.org/10.1002/int.1024

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc

23. Glez-Bedia, M., Corchado, J. M., Corchado, E. S., & Fyfe, C. (2002). Analytical model for constructing deliberative agents. International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications, 10(3).
24. Gonzalez-Briones, A., Chamoso, P., De La Prieta, F., Demazeau, Y., & Corchado, J. M. (2018). Agreement Technologies for Energy Optimization at Home. Sensors (Basel), 18(5), 1633-1633. doi:10.3390/s18051633
25. González-Briones, A., Chamoso, P., Yoe, H., & Corchado, J. M. (2018). GreenVMAS: virtual organization-based platform for heating greenhouses using waste energy from power plants. Sensors, 18(3), 861.
26. Gonzalez-Briones, A., Prieto, J., De La Prieta, F., Herrera-Viedma, E., & Corchado, J. M. (2018). Energy Optimization Using a Case-Based Reasoning Strategy. Sensors (Basel), 18(3), 865-865. doi:10.3390/s18030865
27. Hafewa Bargaoui, Olfa Belkahla Driss (2014). Multi-Agent Model based on Tabu Search for the Permutation Flow Shop Scheduling Problem. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 3, n. 1
28. Jorge Agüero, Miguel Rebollo, Carlos Carrascosa, Vicente Julián (2013). MDD-Approach for developing Pervasive Systems based on Service-Oriented Multi-Agent Systems. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 2, n. 3
29. Leonor Becerra-Bonache, M. Dolores Jiménez López (2014). Linguistic Models at the Crossroads of Agents, Learning and Formal Languages. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 3, n. 4
30. Sittón-Candanedo, I., Alonso, R. S., Corchado, J. M., Rodríguez-González, S., & Casado-Vara, R. (2019). A review of edge computing reference architectures and a new global edge proposal. Future Generation Computer Systems, 99, 278-294.
31. Omar Jassim, Moamin Mahmoud, Mohd Sharifuddin Ahmad (2014). Research Supervision Management Via A Multi-Agent Framework. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 3, n. 4
32. Pablo Chamoso, Fernando De La Prieta (2015). Simulation environment for algorithms and agents evaluation. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 4, n. 3
33. Palomino, C. G., Nunes, C. S., Silveira, R. A., González, S. R., & Nakayama, M. K. (2017). Adaptive agent-based environment model to enable the teacher to create an adaptive class. Advances in Intelligent Systems and Computing (Vol. 617). https://doi.org/10.1007/978-3-319-60819-8_3
34. Paula Andrea Rodríguez Marín, Néstor Duque, Demetrio Ovalle (2015). Multi-agent system for Knowledge-based recommendation of Learning Objects. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 4, n. 1
35. Ricardo Silveira, Guilherme Klein Da Silva Bitencourt, Thiago Ângelo Gelaim, Jerusa Marchi, Fernando De La Prieta (2015). Towards a Model of Open and Reliable Cognitive Multiagent Systems: Dealing with Trust and Emotions. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 4, n. 3
36. Rodolfo Salazar, José Carlos Rangel, Cristian Pinzón, Abel Rodríguez (2013). Irrigation System through Intelligent Agents Implemented with Arduino Technology. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 2, n. 3
37. Rodriguez-Fernandez J., Pinto T., Silva F., Praça I., Vale Z., Corchado J.M. (2018) Reputation Computational Model to Support Electricity Market Players Energy Contracts Negotiation. In: Bajo J. et al. (eds) Highlights of Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection. PAAMS 2018. Communications in Computer and Information Science, vol 887. Springer, Cham
38. Román, J. A., Rodríguez, S., & de da Prieta, F. (2016). Improving the distribution of services in MAS. Communications in Computer and Information Science (Vol. 616). https://doi.org/10.1007/978-3-319-39387-2_4
39. Sigeru Omatu, Tatsuyuki Wada, Pablo Chamoso (2013). Odor Classification using Agent Technology. DCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 2, n. 4

ISBN: 978-84-9012-862-6
Colección Aquilafuente: AQ-242
Ediciones Universidad de Salamanca - cc by nc dc