# CREACION DE PAGINAS WEB

## Lenguaje HTML

## CGI, SSI, JavaScript

# Junio-2000

## INDICE

1. LENGUAJE HTML....4
2. REGLAS DEL LENGUAJE....4
3. ESTRUCTURA DE PAGINA HTML....5
4. COMANDOS HTML....6
4.1. Formatos de Texto....6
4.2. Caracteres Especiales....9
4.3. Color en las Páginas....12
4.4. Listas....14
4.5. Tablas....18
4.6. Enlaces o referencias....18
4.7. Imagenes y otros tipos de información....23
4.8. Formularios....25
4.9. Marcos (Frames)....29
5. CGI....33
6. SERVER SIDE INCLUDES....36
6.1. Un contador SSI con Shell....40
7. JAVA....41
7.1. Especificación del lenguaje JAVA....42
7.2. Applet de JAVA....44
7.3. JavaScript....45
8. HERRAMIENTAS Y UTILIDADES....49
8.1. Editores HTML....49
8.2. Animaciones con GIF....49
8.3. Animación de Páginas....50
8.4. Contadores de Páginas....50
8.5. Propagadores de Páginas en Internet....50
9. GUIA DE ESTILO....51

## 1. LENGUAJE HTML

El lenguaje HTML es el usuado para escribir textos hipermedia en Web. Este documento presenta la versión HTML 2.0, que está recogida en la norma RFC1866.

Se crea en 1991, inventado por Tim-Berners-Lee del CERN. Es un sistema de hipertexto concebido como medio de transmisión de información entre físicos del CERN. En 1993 Dan Connelly escribe la primera especificación del lenguaje HTML 1.0. En 1994 se publica la especificación de HTML 2.0, que incorpora formularios y mapas sensibles.

En la actualidad existe la versión HTML 3.0, aunque es soportada por pocos navegadores. Netscape dice que su navegador soporta la versión HTML 3.0, pero en realidad no es compatible 100%, y presenta diferencias como por ejemplo en cuanto a la realización de tablas.

Paralelamente a la versión de HTML 2.0 se ha definió el Common Gateway Interface (CGI), que define la interfaz entre los programas que atienden los formularios y el servidor de WWW.

## 2. REGLAS DEL LENGUAJE

Este lenguaje define una **página WWW** como un simple archivo de **texto**, donde no importan los **tabuladores** ni los **saltos de línea**. Esta página o archivo se puede editar con **cualquier editor** de texto. Y para darle un formato hipertexto se utilizan **comandos** o **etiquetas** (**tag** en inglés) HTML, que se insertan así: ***<etiqueta>texto</etiqueta>***

Al presentar un fichero HTML no se tienen en cuenta las tabulaciones, los saltos de línea ni los espacios en blanco extra. Esto tiene ventajas y desventajas. La principal ventaja es que permite obtener resultados uniformes y de buena presentación de manera bastante fácil. La principal desventaja es que un documento HTML es necesario utilizar **comandos** para evitar que quede todo el texto en una sola línea.

Los **comandos** se encierran entre los caracteres menor (**<**) y mayor (**>**), quedando de la forma **<etiqueta parametro=valor>texto</etiqueta>**. Algunos comandos tienen parámetros con valor: **<etiqueta parametro=valor>texto</etiqueta>**. Y otros comandos no requieren la finalización **</etiqueta>**.

El **texto** puede ser cualquier carácter del "0" al "9", de la "a" a la "z" y de la "A" a la "Z", además los signos de puntuación y aritméticos. Sin embargo no se pueden utilizar los restantes caracteres de la tabla ASCII. Existen tres caracteres especiales:

**<** Mayor que se usa para indicar el comienzo de un comando HTML.

**>** Menor que se usa para indicar el final de un comando HTML.

**&** Ampersand que se usa para escribir caracteres especiales (símbolos matemáticos, comerciales, así como el signo menor que y el mayor que y el propio ampersand).

## 3. ESTRUCTURA DE PAGINA HTML

Existen dos partes fundamentales de un documento HTML, el encabezado y el cuerpo. El documento completo se encierra con las etiquetas ***<HTML>.....</HTML>***. Este comando o etiqueta permite indicar al programa cliente (Netscape/Explorer) la versión de HTML que estamos usando, entre otras cosas.

### - Encabezado

Se inicia mediante el comando ***<HEAD>*** y se termina con ***</HEAD>***. Por lo general se incluye aquí el título del documento, mediante el comando ***<TITLE>...</TITLE>***. Y el directorio base del documento actual, que se usa si las referencias contenidas dentro de él son relativas: ***<BASE HREF="URL">***.

### - Cuerpo

Se inicia mediante el comando ***<BODY>*** y se termina con el comando ***</BODY>***. Este comando acepta numerosos modificadores. Dentro del cuerpo del documento se incluye el hipertexto a presentar. Uno de sus modificadores son los que corresponden al color de fondo de la página, texto y enlaces (véase apartado de Color de Páginas).

Es importante incluir el comando ***<ADDRESS>...</ADDRESS>*** al final del cuerpo del documento (pero dentro de él). En él se escribe el nombre del autor del documento, la organización a la que pertence, su dirección de correo electrónico, y otra información que se considere relevante (por ejemplo, la última fecha de actualización del documento).

## 4. COMANDOS HTML

### 4.1. Formatos de Texto

**- Cabeceras**

El comando ***<Hn>...</HTn>*** se usa para delimitar una cabecera de tamaño **n**. Los textos en HTML poseen seis niveles de encabezados. El nivel de encabezado 1 se usa para las divisiones mayores del texto y es que hace que el texto sea más grande. El nivel de encabezado 6 es el que muestra el texto más pequeño. Todos los comandos de cabecera insertan un salto de línea al final del texto que delimitan.

**<H1>Nivel de encabezado 1</H1>**
**<H2>Nivel de encabezado 2</H2>**
**<H3>Nivel de encabezado 3</H3>**
**<H4>Nivel de encabezado 4</H4>**
**<H5>Nivel de encabezado 5</H5>**
**<H6>Nivel de encabezado 6</H6>**

**- Realce de Texto**

Hay varios comandos que permiten cambiar la apariencia del texto, para realzarlo sobre el resto. Se trata de delimitadores, que indican donde comienza y donde termina el texto que se verá diferente. Estos comandos se pueden mezclar entre sí. Algunos de ellos son:

**Formatos Lógicos**

*<EM>énfasis (Emphasis)</EM>*
*<CITE>citas</CITE>*
*<VAR>valores de variables</VAR>*
***<KBD>teclas</KBD>***
***<CODE>código fuente</CODE>***
***<LISTING>código fuente</LISTING>***
***<SAMP>muestras</SAMP>***
**<STRONG>Gran énfasis (Strong)</STRONG>**

**Formatos Físicos**

**<B>Negrita (Bold)</B>**
*<I>Cursiva (Italic)</I>*
<U>Subrayado (Underline)</U>
<STRIKE>~~Texto tachado~~</STRIKE>
<SUP>Superíndices</SUP>
<SUB>Subíndices</SUB>
<SMALL>Pequeño</SMALL>
<BLINK>Parpadeante</BLINK>
***<TT>Tamano fijo (TypeWriter)</TT>***

**- Párrafo**

El comando ***<P>...</P>*** se usa como delimitador de párrafo en HTML. Inserta automáticamente un salto de línea al final del párrafo, y produce un espaciamiento entre los diferentes párrafos de un documento. Adicionalmente, permite alinear el texto al centro, a la izquierda o a la derecha (***ALIGN=LEFT o RIGHT o CENTER***).

**- Párrafo indentado**

El comando ***<BLOCKQUOTE>...</BLOCKQUOTE>*** se usa para delimitar un párrafo indentado por la izquierda y por la derecha.

- **Comentarios**

Los fragmentos de texto encerrados en el comando ***<!--...-->*** no son presentados por el visualizador al usuario. Son comentarios de la página.

**- Salto de línea**

El comando ***<BR>*** permite hacer un salto de línea. Adicionalmente, permite realizar saltos más largos o más cortos, dejar atrás las imágenes, etc.

**- Tamaño de letra**

Con el comando ***<FONT SIZE=n>...</FONT>*** se puede modificar el tamaño de la letra y conseguir efectos con el texto, como son los textos de letra curvada.

- **Texto preformateado**

El uso de texto preformateado permite crear un espacio para que el cliente pueda interpretar los tabs, saltos de línea y espacios en blanco incluidos en el documento. Además, en el texto preformateado no se interpretan los caracteres especiales HTML : <, > y &.

Para incluir texto preformateado, se usa el comando ***<PRE>...</PRE>***. El tipo de letra usado por el texto preformateado es de tamaño fijo.

**- Texto Centrado**

Para centrar una parte del documento, se delimita lo que se desea centrar mediante el comando ***<CENTER>...</CENTER>***.

- **Texto en una línea**

Para que los visualizadores no presenten el texto con saltos de línea se utiliza el comando ***<NOBR>...</NOBR>***. De esta forma todo el texto entre el inicio y el fin del comando aparecerá en una única línea.

**- Lineas horizontales**

Una línea horizontal es un separador entre párrafos de texto. Se inserta esta línea con el comando ***<HR>***. Este comando acepta modificadores de tamaño (***SIZE=grosor***), longitud (***WIDTH=pixels o %***), alineación (***ALIGN=LEFT o RIGHT o CENTER***) y sombreado (***NOSHADE***).

En un cliente gráfico, aparece como una delgada línea horizontal, en un cliente de texto, aparece como una línea hecha con guiones.

**- Ejemplo básico**

En este ejemplo, se observan todos los aspectos tratados hasta este momento.

```
<HTML>
<HEAD> <TITLE>Curso de HTML - Ejemplo 1</TITLE> </HEAD>
<BODY>
<H1>Ejemplo 1</H1>
<H2>Primera parte</H2>
<P>Usamos el comando párrafo, para contener un texto
independiente.</P>
Cortamos una línea usando BR.<BR>
<H2>Segunda parte</H2>
<P>Marcamos palabras <B>importantes</B>, otras <B><I>mas
importantes</I></B>, y otras <STRONG>muy importantes</STRONG></P>
<H3>Nota</H3>
<P>Tambien hacemos <BLINK>parpadear</BLINK> texto.</P>
<ADDRESS>
alumno@mailhost.servidor.es<BR>
La Coru&ntilde;a, 1 de Enero de 1997<BR>
</ADDRESS>
</BODY> </HTML>
```

### 4.2. Caracteres Especiales

Los caracteres especiales constituyen una gran ayuda para quienes escribimos textos en español, pues permiten generar las vocales acentuadas: "á", "é","í", "ó","ú", y la letra ñ. Para incluir un caracter especial, se usa el comando ***&nombre_entidad*;** o bien ***&#numero_entidad*;**. Es mejor utilizar el nombre de entidad en vez del número, por motivos de claridad. RFC1866 también recomienda esto. El número de entidad es el número de caracter ISO Latin-1 o ISO-8859-1.

Si en el visualizador de HTML no se muestra el carácter, sino que aparece el nombre de la entidad, entonces el visualizador no reconoce el elemento. Sin embargo si no aparece nada es posible que el visualizador HTML sí lo reconozca, sin embargo no se ve, porque se está utilizando un tipo de fuente de letra que no lo soporta.

Los códigos de los caracteres especiales necesarios para el español se muestran en la tabla siguiente, donde aparece el nombre de la entidad y/o el número, y después el carácter tal como debería verse y su descripción:

| Nombre Entidad | Núm. Entidad | Carácter | Descripción |
|---|---|---|---|
| &aacute; | &#224; | á | a con acento agudo |
| &eacute; | &#232; | é | e con acento agudo |
| &iacute; | &#236; | í | i con acento agudo |
| &oacute; | &#242; | ó | o con acento agudo |
| &uacute; | &#249; | ú | u con acento agudo |
| &Ntilde; | &#209; | Ñ | |
| &ntilde; | &#241; | ñ | |

Existen muchos caracteres especiales y símbolos matemáticos. en la siguiente tabla aparece una extensa tabla de estos caracteres.

| Nombre Entidad | Núm. Entidad | Carácter | Descripción |
|---|---|---|---|
| | &#00;-&#08; | | No usado |
| | 	 | | Tabulador |
| | 
 | | Nueva Línea |
| | &#11;-&#31; | | No usado |
| |   | | Espacio |
|   | | | Espacio |
| ! | ! | ! | Cierre de exclamación |
| " | " | " | Dobles comillas |
| | # | # | Ventana |
| | $ | $ | Dolar |
| | % | % | Porcentaje |
| & | & | & | Ampersán |
| | ' | ' | Apóstrofo |
| | ( | ( | Abrir paréntesis |
| | ) | ) | Cerrar paréntesis |
| | * | * | Asterisco |
| | + | + | Signo más |
| | , | , | Coma |
| | - | - | Signo menos |
| | . | . | Punto |
| ÷ | / | / | Signo de dividir |
| | 0-9 | | Digitos 0 a 9 |
| | : | : | Dos puntos |
| | ; | ; | Punto y coma |
| < | < | < | menor que |
| | = | = | signo igual |
| > | > | > | mayor que |
| ? | ? | ? | Cierre de pregunta |
| | @ | @ | arroba (at) |
| | A-Z | | Letras A-Z |
| | [ | [ | Abrir corchetes |
| | \ | \ | Divivisión invertida |
| | ] | ] | Cerrar corchetes |
| | ^ | ^ | Acento circunflejo |
| | _ | _ | Subrayado |
| | ` | ` | Acento agudo |
| | a-z | | Letras a-z |
| | { | } | Abrir llaves |
| | | | \| | Barra vertical |
| | } | { | Cerrar llaves |
| | ~ | ~ | Tilde |
| | &#127;-  | | No usados |
| ¡ | ¡ | ¡ | Abrir exclamacion |
| | ¢ | ¢ | Centavo |
| £ | £ | £ | Libra esterlina |
| | ¤ | ¤ | Moneda corriente |
| ¥ | ¥ | ¥ | Yen |
| ¦ | ¦ | ¦ | Barra vertical partida |
| § | § | § | Sección |
| ¨ | ¨ | ¨ | Diéresis |
| © | © | | Copyright |
| ª | ª | ª | Signo Femenino |
| « | « | « | Mucho menor |
| ¬ | ¬ | ¬ | Signo No |
| |  | | Guión |
|  | | - | Guión |
| ® | ® | ® | Marca Registrada |
| ¯ | ¯ | ¯ | Suprarrayado |
| ° | ° | º | Grado |
| ± | ± | ± | Mas o menos |

| | | | |
|---|---|---|---|
| ² | ² | ² | Superindice 2 |
| ³ | ³ | ³ | Superindice 3 |
| | ´ | ´ | Acento agudo |
| µ | µ | µ | Signo Micro |
| ¶ | ¶ | ¶ | Párrafo |
| · | · | · | Punto medio |
| ¸ | ¸ | ¸ | Cedilla |
| ¹ | ¹ | ¹ | Superindice 1 |
| º | º | º | Masculino |
| » | » | » | Mucho menor |
| ¼ | ¼ | ¼ | Fraccion un cuarto |
| ½ | ½ | ½ | Fraccion un medio |
| ¾ | ¾ | ¾ | Fraccion tres cuartos |
| ¿ | ¿ | ¿ | Abrir pregunta |
| Á | À | À | A con acento agudo |
| À | Á | Á | A con acento grave |
| Â | Â | Â | A acento circunflejo |
| Ã | Ã | Ã | A con tilde |
| Ä | Ä | Ä | A con diéresis |
| Å | Å | Å | A con anillo |
| Æ | Æ | Æ | AE (diptongo) |
| Ç | Ç | Ç | C con cedilla |
| É | È | È | E con acento agudo |
| È | É | É | E con acento grave |
| Ê | Ê | Ê | E acento circunflejo |
| Ë | Ë | Ë | E con diéresis |
| Í | Ì | Ì | I con acento agudo |
| Ì | Í | Í | I con acento grave |
| Î | Î | Î | I acento circunflejo |
| Ï | Ï | Ï | I con diéresis |
| Ð | Ð | Ð | Eth |
| Ñ | Ñ | Ñ | |
| Ó | Ò | Ò | O con acento agudo |
| Ò | Ó | Ó | O con acento grave |
| Ô | Ô | Ô | O acento circunflejo |
| Õ | Õ | Õ | O con tilde |
| Ö | Ö | Ö | O con diéresis |
| × | × | × | Signo de multiplicar |
| Ø | Ø | Ø | O cruzada |
| Ú | Ù | Ù | U con acento agudo |
| Ù | Ú | Ú | U con acento grave |
| Û | Û | Û | U acento circunflejo |
| Ü | Ü | Ü | U con diéresis |
| Ý | Ý | Ý | Y con acento agudo |
| Þ | Þ | Þ | THORN |
| ß | ß | ß | Beta |
| á | à | á | a con acento agudo |
| à | á | à | a con acento grave |
| â | â | â | a acento circunflejo |
| ã | ã | ã | a con tilde |
| å | ä | ä | a con diéresis |
| ä | å | å | a con anillo |
| æ | æ | æ | ae (diptongo) |
| ç | ç | ç | c con cedilla |
| é | è | é | e con acento agudo |
| è | é | é | e con acento grave |
| ê | ê | ê | e con circunflejo |
| ë | ë | ë | e con diéresis |
| í | ì | í | i con acento agudo |
| ì | í | ì | i con acento grave |
| î | î | î | i con circunflejo |

| | | | |
|---|---|---|---|
| ï | ï | ï | i con diéresis |
| ð | ð | ð | eth |
| ñ | ñ | ñ | |
| ó | ò | ó | o con acento agudo |
| ò | ó | ò | o con acento grave |
| ô | ô | ô | o con circunflejo |
| õ | õ | õ | o con tilde |
| ö | ö | ö | o con diéresis |
| | ÷ | ÷ | Signo de división |
| ø | ø | ø | o cruzada |
| ú | ù | ú | u con acento agudo |
| ù | ú | ù | u con acento grave |
| û | û | û | u con circunflejo |
| ü | ü | ü | u con diéresis |
| ý | ý | ý | y con acento agudo |
| þ | þ | þ | thorn |
| ÿ | ÿ | ÿ | y con diéresis |
| ™ | | ™ | TradeMark |

### 4.3. Color en las Páginas

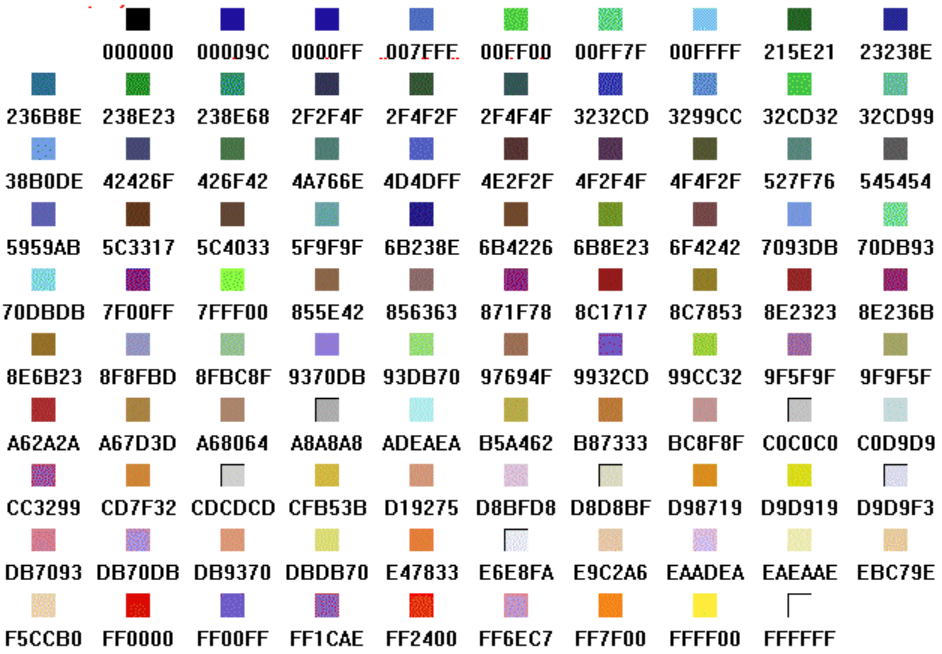
Al iniciar el cuerpo de la página se puede indicar el color que ha de mostrar mediante atributos del comando ***<BODY>***. Estos atributos son:

- ***BGCOLOR***: (Background Color) o color de fondo de la página.
- ***TEXT***: color del texto normal.
- ***LINK***: color de los enlaces no visitados.
- ***VLINK***: (visited link) color de los enlaces visitados.
- ***ALINK***:(active link) color que adquiere un enlace cuando se pulsa con el ratón.
- ***BACKGROUND***: URL de la imágen usada para el fondo de la página.

Ejemplo:

```
<BODY BACKGROUND=fondoam.gif BGCOLOR=#C0FFFF TEXT=#000000 LINK=#FF0000
VLINK=#408000 ALINK=#400080>
```

Los colores están en formato RGB, de tal forma que el rojo es ***FF0000***, el verde ***00FF00***, y el azul es ***0000FF***.


000000 00009C 0000FF 007FFF 00FF00 00FF7F 00FFFF 215E21 23238E
236B8E 238E23 238E68 2F2F4F 2F4F2F 2F4F4F 3232CD 3299CC 32CD32 32CD99
38B0DE 42426F 426F42 4A766E 4D4DFF 4E2F2F 4F2F4F 4F4F2F 527F76 545454
5959AB 5C3317 5C4033 5F9F9F 6B238E 6B4226 6B8E23 6F4242 7093DB 70DB93
70DBDB 7F00FF 7FFF00 855E42 856363 871F78 8C1717 8C7853 8E2323 8E236B
8E6B23 8F8FBD 8FBC8F 9370DB 93DB70 97694F 9932CD 99CC32 9F5F9F 9F9F5F
A62A2A A67D3D A68064 A8A8A8 ADEAEA B5A462 B87333 BC8F8F C0C0C0 C0D9D9
CC3299 CD7F32 CDCDCD CFB53B D19275 D8BFD8 D8D8BF D98719 D9D919 D9D9F3
DB7093 DB70DB DB9370 DBDB70 E47833 E6E8FA E9C2A6 EAADEA EAEAAE EBC79E
F5CCB0 FF0000 FF00FF FF1CAE FF2400 FF6EC7 FF7F00 FFFF00 FFFFFF

### 4.4. Listas

Existen varios tipos de lista. Todas permiten varios niveles; es decir, podemos tener una lista dentro de otra.

**- Listas descriptivas**

Una lista descriptiva es muy útil al momento de hacer glosarios, índices, referencias a otros documentos, o describir, en general, cualquier cosa. Se declaran usando el comando ***<DL>...</DL>***. Cada ítem de la lista consta de un título, marcado mediante ***<DT>...</DT>***, y una descripción para dicho título, que se indica mediante el comando ***<DD>***. El comando ***<DL>*** admite el modificador ***COMPACT***, que hace que los items se presenten en una línea. Una lista descriptiva se ve así:

Cosas
- Minerales
  - Metamórficos
  - Sedimentarios
  - Volcánicos

Animales
- Mamíferos
- Aves
- Peces

Y en HTML se escribe:

```
<DL>
<DT>Cosas
<DL>Minerales
<DD>Metam&oacute;rficos
<DD>Sedimentarios
<DD>Volc&aacute;nicos
</DL>
<DT>Animales
<DD>Mam&iacute;feros
<DD>Aves
<DD>Peces
</DL>
```

**- Listas numeradas**

También llamadas listas ordenadas, se declaran usando el comando ***<OL>...</OL>***. Cada item de la lista se marca con un ***<LI>***. El resultado final es que el item se separa del borde izquierdo y aparece marcado con un número. Este comando permite especificar el tipo de numeración que se dará con el cualificador ***TYPE=A,a,I,i,1*** (letras mayúsculas, minúsculas, números romanos en mayúsculas, en minúsculas y números arábigos). Además el cualificador ***START=n*** indica donde comienzan las líneas. Una lista numerada se ve así:

1. Capítulo 1
   1. Tema 1
   2. Tema 2
   3. Tema 3
2. Capítulo 2
   1. Tema 1
3. Capítulo 3
   1. Tema 1
   2. Tema 2

Y en HTML se escribe:

```
<OL>
<LI> Cap&iacute;tulo 1
        <OL>
        <LI>Tema 1
        <LI>Tema 2
        <LI>Tema 3
        </OL>
<LI> Cap&iacute;tulo 2
        <OL>
        <LI>Tema 1
        </OL>
<LI> Cap&iacute;tulo 3
        <OL>
        <LI>Tema 1
        <LI>Tema 2
        </OL>
</OL>
```

**- Listas no numeradas**

También llamadas listas no ordenadas, se declaran usando el comando ***<UL [TYPE="disc/circle..."]>...</UL>***. Cada item de la lista se marca con una ***<LI>***. El resultado final es que el item se separa del borde izquierdo y aparece marcado con un símbolo (un punto, un cuadrado, un círculo etc.. depende del nivel y del atributo *TYPE*). Una lista no numerada se ve así:

· España
    Madrid
· Italia
    Roma
· Francia
    Paris

Y en HTML se escribe:

```
<UL>
<LI>Espa&ntilde;a
<UL>
<LI>Madrid
</UL>
<LI>Italia
<UL>
<LI>Roma
</UL>
<LI>Francia
<UL>
<LI>Paris
</UL>
<LI> ...
</UL>
```

**- Listas de directorio**

Son similares a las listas no numeradas, en uso (también utilizan el comando ***<LI>*** para los item) y en presentación. Se construyen con el comando ***<DIR>...</DIR>***.

**- Listas de menú**

También son similares a las listas no numeradas. Se construyen con el comando ***<MENU>...</MENU>***.

## - Ejemplo

```
<HTML>
<HEAD><TITLE>Curso de HTML - Ejemplo 2</TITLE></HEAD>
<BODY>
<H2>Listas</H2>
<H3>No numeradas</H3>
<OL><LI>Usamos litas no numeradas
<LI>Para clasificar la información
</OL>
<H3>Numeradas</H3>
<UL><LI>Podemos hacer que HTML
<LI>cuente por nosotros
</UL>
<HR>
<H2>Descripciones</H2>
<DL>
<DT>Descripcion</DT>
<DD>Es nombrar las propiedades de algo
<DT>Lista descriptiva</DT>
<DD>Es nombrar las propiedades de varios objetos
</DL>
<CENTER>También sabemos centrar el texto.</CENTER>
<TT><PRE> y escribir preformateado </PRE></TT>
</BODY></HTML>
```

### 4.5. Tablas

Una tabla es una manera muy compacta y clara de mostrar la información. Una tabla en HTML se entiende como un conjunto de filas (fila=horizontal), apiladas una sobre otra. Cada fila contiene a su vez un conjunto de celdas, puestas una al lado de la otra.

Una tabla se declara usando el comando ***<TABLE>...</TABLE>***. Dentro de la tabla, se usa ***<TR>...</TR>*** para indicar una fila y dentro de una fila, ***<TD>...</TD>*** para delimitar el contenido de una celda normal y ***<TD>...</TD>*** para delimitar el contenido de una celda de cabecera o negrita. El elemento de cierre de fila y de celda es opcional. Además para asignar un título a esta tabla se pone dentro el comando ***<CAPTION [ALIGN=top/bottom>...</CAPTION>***.

El comando ***TABLE*** acepta los atributos siguientes:
- ***BORDER***: para indicar el tamaño del borde de la tabla.
- ***CELLSPACING***: es el número de espacios entre cada celda.
- ***CELLPADING***: es el número de espacios entre el contenido y el borde.
- ***WIDTH***: indica el porcentaje o los pixeles de pantalla que ocupa la tabla de ancho.
- ***HEIGHT***: indica el porcentaje o los pixeles de pantalla que ocupa una celda de alto.
- ***BORDER***: para indicar el tamaño del borde de la tabla.

El comando ***TR*** acepta los atributos siguientes:
- ***ALIGN***: indica la alineación a derecha o izquierda del texto dentro de la celda (*left*, *rigth*).
- ***VALIGN***: indica la alineación arriba o abajo del texto dentro de la celda (*top*, *bottom*, *middle*, *baseline*).

Los comandos ***TD*** y ***TH*** aceptan los atributos ***ALIGN*** y ***VALIGN*** como ***TR*** y además:
- ***COLSPAN***: indica el número de celdas verticales que ocupa ésta.
- ***ROWSPAN***: indica el número de celdas horizontales que ocupa ésta.

De este modo se pueden crear tablas con múltiples formas. Un ejemplo sencillo.

```
<TABLE BORDER=3>
<TR><TH></TH><TH>Varones</TH><TH>Hembras</TH>
<TR><TH>Peso Medio</TH><TD>80</TD><TD>60</TD>
<TR><TH>Altura<TD>170<TD>160
</TABLE>
```

### 4.6. Enlaces o referencias

La capacidad de hacer referencias en un hipertexto, es sin duda su capacidad más atrayente y práctica para poder entregar de mejor manera la información. Las referencias o enlaces se definen mediante los **URL** (Uniform Resource Locator) que constituyen una manera estandarizada de dar una dirección en Internet a cualquier recurso. Los enlaces (*links*) pueden ser de varios tipos:

- Referencia a otra zona dentro del mismo documento.
- Referencia a otro documento.
- Referencia a otro ordenador.
- Referencia a otro servicio.

Para entender el concepto de enlaces en HTML, hay que definir previamente lo que es una **URL** (Uniform Resource Location). Este concepto puede verse como una extensión al sistema de ficheros del ordenador local, que incluye a todos los documentos que están en la red.

Una URL tiene 3 partes:

- El **protocolo** por el que se accede al documento (*httpd*, *ftp*), y el **puerto**.
- La **máquina** donde está el documento (descrita por su nombre y dominios)
- El **directorio** y **nombre** del documento en esa máquina (generalmente relativo)

Por ejemplo:

*http://www.empresa.es/index.html:3080*
*file://usr/include/stdio.h*

Para crear enlaces se tiene el comando **<A>** que puede tomar varios parámetros. La forma básica de hacerlo es con el siguiente parámetro: **<A HREF="URL">...</A>**. Todo lo que esté encerrado por la etiqueta se convierte en un enlace a esa URL, por lo que si encierra una imagen será también un enlace.

- **Referencia dentro de un mismo documento**

Es posible hacer referencias que lleven al usuario de una parte del documento a otra dentro del mismo documento con solo pulsar el ratón. Llamaremos a estas referencias *referencias locales*. Estas referencias son muy útiles para crear índices al principio del documento, y constan de dos objetos:

- La **referencia** que corresponde a la zona en la cual el usuario pulsa el ratón, para moverse a otra parte del documento. Esta zona aparece subrayada en el documento, y se crea delimitando la zona con el comando **<A HREF="#nombre">...</A>**; en donde nombre es la etiqueta que se está referenciando.

- El **nombre referenciado** que corresponde a la zona del documento a la cual el usuario llegará al pulsar el ratón en la referencia correspondiente. Un nombre local se crea delimitando la zona con el comando **<A NAME="nombre">...</A>**; en donde "nombre" es la etiqueta que asignamos a la zona.

**- Ejemplo**

```
<HEAD><TITLE>Ejemplo 3. Referencias locales</TITLE>
</HEAD> <BODY>
<H2>Referencias locales</H2>
<H3><A NAME="indice">Indice</A></H3>
<UL> <LI><A HREF="#uno">Sección uno</A>
<LI><A HREF="#dos">Sección dos</A>
<LI><A HREF="#tres">Sección tres</A>
</UL>
<H3><A NAME="uno">Sección uno</A></H3>
<P>Pulse aqui para <A href="#indice">volver al índice</A>
<H3><A NAME="dos">Sección dos</A></H3>
<H3><A NAME="tres">Sección tres</A></H3>
</BODY>
```

**- Referencia a otro documento**

Para referenciar a otro documento, es necesario conocer la ubicación exacta del documento que se referenciará. Una ubicación, puede ser referenciada en forma *relativa* o *absoluta*

- **Ubicación relativa** Se indica especificando la posición del documento en la estructura de subdirectorios a partir de la ubicación del documento actual. **Sólo se puede usar para documentos ubicados en el mismo ordenador.** Una referencia relativa a otro documento se hace usando el comando ***<A HREF="ubicación">...</A>***.

- **Ubicación absoluta** Se indica especificando el URL de la página que se está referenciando. Los URL son una manera universal de especificar una dirección. La fórma más básica de referenciar un hipertexto en este modo es usando el comando:
***<A HREF="http://ordenador/directorio/archivo">...</A>***. El *ordenador* indica la máquina (y dominio) donde se encuentra el documento. El *directorio* y *archivo* indican su posición dentro de ese ordenador.

**- Ejemplo**

```
<HEAD><TITLE>Ejemplo 4. Referencias a otros documentos</TITLE>
</HEAD>
<BODY>
<H1>Referencias a otros documentos</H1>
<H2>Referencias relativas</H2>
<P>Se puede referenciar, por ejemplo, a un archivo localizado en el
mismo directorio, como por ejemplo, al <AHREF="ejemplo1.htm">ejemplo
1</A>.
Tambi&eacute;n a un archivo localizado en otro directorio, por
ejemplo, el texto acerca de los
<A HREF="../ejemplos/url.htm">URLs</a></P>
<H2>Referencias absolutas</H2>
<H3>Algunos servicios WWW en Espa&ntilde;a</H3>
<P>En Espa&ntilde;a tenemos el servicio <A
HREF="http://www.ole.es/">OLE</A> que es un buscador.<BR>
<H3>Algunos servicios en el extranjero</H3>
<P>Para informarse de la actualidad mundial, se puede acceder al
servicio <A HREF="http://www.cnn.com/">CNN</A>. Para los amantes de
la música se encuentra el servidor de <A
HREF="http://mtv.com/">MTV</A> y para localizar distintos tipos de
información está <A HREF="http://www.yahoo.com/">¡Yahoo!</A>.
</BODY>
```

- **Referencia a otro ordenador**

Para este tipo de referencias se indica el URL del ordenador y del documento a referenciar. Incluso se puede añadir el signo # y luego el nombre de la zona a referenciar dentro del documento. Es una combinación de los anteriores.

- **Referencia a otro servicio**

Los servicios son los distintos canales de conexión entre servidor y cliente utilizando TCP/IP. Los servicios más comunes son WWW, FTP, E-mail y News. Cada recurso de la red Internet está completamente definido por el servicio, el ordenador, el directorio/buzón/grupo, y el fichero/artículo/mensaje. De esta forma vamos a ver como es posible refenciar recursos de diferentes servicios.

**. Referencia a un hipertexto**

**Sintaxis:** *http://ORDENADOR:PUERTO/DIR1/.../DIRn/ARCHIVO#SECCION*

Ejemplo: http://www.empresa.es/sucursales/index.html

El URL comienza con *http://*, que indica que es una referencia a un hipertexto. A continuación se indica el nombre del *ordenador* en donde se encuentra el recurso, después el *DIRECTORIO*, y finalmente el nombre del *ARCHIVO* del hipertexto. Opcionalmente se puede añadir la *SECCION* o parte del documento.

El *PUERTO* es opcional, e indica el número de canal TCP/IP que está atendiendo el proceso del servidor que aporta el servicio WWW. Cuando no se especifica el nombre del *ARCHIVO*, el servidor aportará uno por defecto (normalmente uno llamado *index.html*). Así, en la mayoría de los casos, sólo es necesario el nombre del ordenador.

El directorio que se indica está tomado en referencia al directorio base del servicio HTML (no coincide con la raíz del árbol de subdirectorios del computador).

Se puede indicar el directorio de un usuario. En este caso, tampoco se accederá al directorio *HOME* del usuario, sino a un subdirectorio llamado *pub_www* localizado a partir del directorio *HOME* del usuario.

**. Referencia a un servicio FTP**

**Sintaxis:** *ftp://***[***user:pass@***]***ordenador:puerto/DIR1/.../DIRn/ARCHIVO*

Ejemplo: *ftp://ftp.empresa.es/pub/*

El URL comienza con *ftp://*, que indica que es una referencia a un servicio de transferencia de ficheros. A continuación, se indica el nombre del *ORDENADOR* que ofrece el servicio FTP (opcionalmente se puede poner previamente el usuario *user* y su contraseña *passw@*), después el *DIRECTORIO* en que se encuentra, y finalmente el nombre del *ARCHIVO*. En caso de que no se especifique *ARCHIVO*, se estará referenciando el *DIRECTORIO* completo. Si no se especifica el *DIRECTORIO*, se referenciará a la raíz del servicio FTP. El *PUERTO* tiene la misma función que en el punto anterior.

**. Referencia a un archivo**

**Sintaxis:** *file://ORDENADOR:PUERTO/DIR1/.../DIRn/ARCHIVO*

Ejemplo: *file://www.empresa.es/pub/readme*

El URL comienza con *file://*, que indica que es una referencia a un archivo cualquiera. A continuación se indica el nombre del *ORDENADOR* donde se encuentra el archivo, después el *DIRECTORIO*, y finalmente el nombre del *ARCHIVO*. En este caso, siempre debe especificarse el nombre del archivo.

**. Referencia a un grupo de noticias**
**Sintaxis:** *news:GRUPO*
Ejemplo: *news:alt.dev.null*
El URL comienza con *news:*, que indica que es una referencia a un grupo de noticias. A continuación se indica el nombre del *GRUPO* de noticias.
La forma en que el cliente recibirá la información del grupo de noticias depende exclusivamente de él. Es decir, debe estar adecuadamente configurado para leer noticias desde algún servidor. No es posible, por lo tanto, especificar en el URL desde que ordenador deberá tratar de obtener la información del grupo de News.

**. Referencia a un fichero de grupo de noticias**
**Sintaxis:** *newsrc:DIR1/.../DIRn/ARCHIVO*
Ejemplo: *news:dirnoti/grupostrabajo/CADCAM*
El URL comienza con *newsrc:*, que indica que es una referencia a un fichero que contiene un determinado grupo de noticias. A continuación se indica el fichero.

**. Referencia a un fichero de grupo de noticias de otro Servidor**
**Sintaxis:** *nntp://ORDENADOR:PUERTO/GRUPO*
Ejemplo: *nntp://rediris.es/alt.medicina*
El URL comienza con *nntp:*, que indica que es una referencia a un servidor de News, que puede ser distinto al que tiene el usuario configurado. A continuación se indica el *ordenador*, y el *grupo* de noticias.

**. Referencia a una dirección e-mail**
**Sintaxis:** *mailto:USUARIO1,USUARIO2,...,USUARIOn*
Ejemplo: mailto:pepe@empresa.es
El URL comienza con *mailto:*, que indica que es una referencia a uno o varios *USUARIOS* de correo electrónico. A continuación se indica el nombre o nombres de los *USUARIOS* que recibirán el correo electrónico.
El cliente debe estar adecuadamente configurado para poder enviar el correo.

### 4.7. Imagenes y otros tipos de información

Para hacer más atractivo y fácil de entender un hipertexto, es posible incluir imágenes, sonido y animaciones. Las imágenes aparecen directamente en el hipertexto (en los casos en que el cliente tenga de una interfaz gráfica); en cambio, los sonidos y las animaciones exigen programas específicos para ser vistos y oidos.

- **Imagenes**

Para incluir una imágen en un hipertexto es necesario utilizar el comando ***<IMG SRC="imagen" ALT=. ALIGN=. HEIGTH=. WIDTH=. BORDER=. ISMAP USEMAP='mapa.html'>*** que no tiene terminador. En donde la *imagen* puede estar referenciada en forma absoluta o relativa, de la misma forma en que se referenciaba a los documentos hipertexto en la sección anterior.

Las imagenes pueden estar en formato GIF o JPG (JPEG). Cuando sean fotografías es preferible el formato JPG, en cualquier otro caso son preferibles en formato GIF. No son recomendables ficheros de imagenes con tamaño superior a 50 Kb.

El comando ***<IMG>*** acepta múltiples modificadores, como por ejemplo, *align* que especifica la alineación de la imagen (*top, texttop, middle, absmiddle, bottom, absbottom, baseline, left, right*) en la línea de texto, *alt* que especifica un texto a presentar si no se presenta la imagen (***ALT=texto***), otros permiten especificar el alto(***HEIGTH=pixels***), el ancho(***WIDTH=pixels.***), el borde (***BORDER=n***), y si es un mapa sensible (***ISMAP***), o si utiliza una descripción de coordenadas para el mapa sensible (***USEMAP='mapa.html'***)

Los mapas sensibles permiten generar páginas con enlaces en las imágenes. Usando ***ISMAP*** cuando el usuario pulsa sobre la imagen con el ratón se le pide al servidor la página referenciada con las coordenadas en la imagen de la pulsación. Usando ***USEMAP*** si el usuario pulsa sobre un punto de la imagen de un polígono definido en el mapa se le pide al servidor la página que corresponde a esa región. El mapa puede estar dentro del mismo documento. Sin embargo este método no es soportado por todos los visualizadores. Ejemplo:

```
<IMG SRC="../imagenes/dibujo.gif" USEMAP="mapas.html#mapa"></A>
```

Ambos métodos se pueden utilizar combinados. Ejemplo:

```
<A HREF="/cgi-bin/imagen"><IMG SRC="../imagenes/dibujo.gif"
USEMAP="mapas.html#mapa" ISMAP></A>
```

Un mapa y sus regiones se describen con el comando ***MAP*** de la siguiente forma:

```
<MAP NAME="name">
<AREA [SHAPE="shape"] COORDS="x,y,..." [HREF="URL"] [NOHREF]>
.................
<AREA [SHAPE="shape"] COORDS="x,y,..." [HREF="URL"] [NOHREF]>
</MAP>
```

Esta descripción de mapa puede estar contenida en la misma página. Cada región contiene sus coordenadas y su referencia. El atributo ***NAME*** indica el nombre del mapa refenciado en el comando ***IMG*** cuando está en la misma página. El atributo ***SHAPE*** indica el tipo de región (actalmente solo está definido el rectángulo *RECT*). El atributo ***COORDS*** son las coordenadas de la región en pixels de la imagen (izquierda, arriba, derecha, abajo). El punto superior izquierda de una imagen es el punto (0,0). El atributo ***NOHREF*** indica que esta región no contiene enlace. El atributo ***HREF*** indica el enlace (otra página, FTP, ...) referenciado al pulsar sobre la región. Si dos regiones interseccionan y el usuario pulsa en la intersección se solicita la primera región definida.

Ejemplo completo:

```
<HTML>
......
<MAP NAME="botones">
<AREA SHAPE="RECT" COORDS="10,10,49,49" HREF="empresa.html">
<AREA SHAPE="RECT" COORDS="60,10,99,49" HREF="producto.html">
<AREA SHAPE="RECT" COORDS="110,10,149,49" HREF="servicio.html">
<AREA SHAPE="RECT" COORDS="160,10,199,49" HREF="indice.html">
</MAP>
......
<IMG SRC="../images/barra.gif" USEMAP="#botones">
......
</HTML>
```

Los visualizadores que no soportan este tipo de regiones en imagenes, ignorarán las descripciones del mapa.

La posibilidad de tener la descripción del mapa en un fichero externo, permite tener la misma descripción del mapa para múltiples páginas, y si hay que cambiar este mapa general para todas las páginas, solamente se ha de modificar este fichero.

- **Sonido**

Los archivos de sonidio son, por lo general, bastante grandes en cuanto al espacio en disco que ocupan; por esto mismo, son también bastante lentos de transferir. Es por esto que los archivos de audio no se suelen cargar de manera automática, sino que es preferible transmitirlos solo cuando el usuario lo solicita. Para incluir un sonido, se delimita la sección con un comando que apunte al **URL** del **archivo de sonido**, de la misma forma que si fuese un documento o una imagen. Aunque, también es posible poner un sonido de fondo a una página, pero la forma de hacerlo depende del visualizador (Netscape o Explorer).

- **Animaciones y video**

Para incluir una animación, se procede de la misma manera que para audio, apuntando al URL del archivo de la animación. Aunque en este caso existen muchas otras alternativas mediante los *Plug-In* de los visualizadores (video interactivo -InLine video-, realidad virtual -VRML-, ...). Esta facilidad está actualmente cambiando constantemente, y parte no se aborda en este documento.

### 4.8. Formularios

Un formulario es una opción de HTML que permite crear páginas donde el usuario puede introducir información para enviar al servidor.

Las aplicaciones de los formularios son muy variadas: solicitar un producto, enviar comentarios, hacer una operacion bancaria, inscribirse en una base de datos, etc.

Un formulario se define con el comando: ***<FORM ACTION="...">*** que apunta a un programa ejecutable que lo procesa, con botones para enviar y limpiar el formulario, y con campos de entrada de datos. Al final de la parte del formulario se pone el comando ***</FORM>***. Todos los elementos del formulario deben estar dentro del *FORM*, o de lo contrario son ignorados.

Cuando el usuario pulsa el botón de enviar el formulario (después de haberlo rellenado), se ejecuta en el servidor el programa ejecutable, el cual recibe como parámetros los campos de entrada del formulario de la forma *NOMBRE=valor*, en que *NOMBRE* es el ***NAME*** identificador del campo de entrada, y *valor* es lo introducido por el usuario. La información acerca de los programas ejecutables está en un capítulo posterior.

El cuerpo del formulario puede contener varios tipos de campos de entrada: entrada de texto (de una línea, o multilínea), opciones seleccionables por medio de un menú desplegable, o por un grupo de opciones, donde solo se permite seleccionar una, y también donde se permiten seleccionar varias.

Todos los campos de entrada tienen asociado un *NOMBRE*, que es **obligatorio** asignar.

**- Entrada de texto**

Para **una línea** de texto, se define mediante: ***<INPUT NAME="nombre" TYPE=TEXT>***. Adicionalmente se puede especificar el tamaño (***SIZE***) de la caja de entrada, en caracteres.

Para **varias líneas**, se define mediante: ***<TEXTAREA NAME="nombre" ROWS="lineas" [WRAP=..] COLS="columnas">...</TEXTAREA>***. No es necesario que haya contenido dentro del TEXTAREA, de hecho en la mayoría de los casos, se deja vacío. El atributo *WRAP* indica la forma en que se cortan las líneas al introducir el texto (*OFF* indica que no se cortan automáticamente, *VIRTUAL* indica que se cortan al mostrarlas al usuario, pero no se envía cortadas al servidor y *PHYSICAL* indica que sí se cortan y envían cortadas al servidor). El atributo *WRAP* es específico de Netscape.

Admite los cualificadores ***MAXLENGTH=nn*** para limitar el número de caracteres que el usuario puede introducir en el campo,

**- Entrada oculta**

Es una entrada que no se muestra al usuario. Se define mediante: ***<INPUT NAME="nombre" TYPE=HIDDEN>***.

**- Entrada de contraseña**

Es una entrada igual a las de texto pero que por cada letra que el usuario introduce, en pantalla aparecen asteriscos. Se define mediante: ***<INPUT NAME="nombre" TYPE=PASSWORD>***.

**- Entrada sobre Imagen**

Es un campo de entrada especial. Muestra una imagen, y cuando el usuario pulsa sobre ella se envía al Servidor el Formulario con las coordenadas pulsadas. Se crea con el comando: ***<INPUT SRC= "imagen" NAME="nombre" TYPE=IMG [ALIGN=..]>***.

**- Entrada de una opción**

La primera alternativa para que el usuario pueda elegir una sola opción entre varias es el uso de **botones** (*radio buttons*). Esto se define mediante una serie de comandos ***<INPUT TYPE=RADIO NAME="nombre" VALUE="valor">***, donde todos comparten el mismo nombre pero tienen distintos valores. En el momento del envío de la respuesta del formulario, lo que se envía es el nombre, y el valor de la opción que fue elegida. Ejemplo:

```
<P:>Sexo:</P>
<INPUT TYPE=RADIO NAME="sexo" VALUE="masc"> Masculino <BR>
<INPUT TYPE=RADIO NAME="sexo" VALUE="fem"> Femenino <BR>
```

La opción que aparece seleccionada por defecto se marca con el atributo *CHECKED* (ejemplo: ***<INPUT TYPE=RADIO NAME="sexo" VALUE="fem" CHECKED> Femenino***). Si no se especifica, ninguna aparecerá seleccionada.

Otra forma de que el usuario pueda seleccionar una opción entre varias es usar un **menú desplegable**. Se define con el comando ***<SELECT>***, seguido de un conjunto de valores con el comando ***<OPTION VALUE>*** que corresponde a las opciones. Ejemplo:

```
Sexo:
<SELECT NAME="sexo">
<OPTION VALUE="masc"> Masculino
<OPTION VALUE="fem"> Femenino
</SELECT>
```

La opción que aparecerá seleccionada se marca dándole el atributo *SELECTED* (ejemplo: ***<OPTION VALUE="fem" SELECTED> Femenino***). Si no se especifica, ninguna aparecerá seleccionada.

El comando *SELECT* permite además, que el usuario seleccione varias opciones a la vez, mediante el atributo *MULTIPLE*. El comando *OPTION* también el atributo *DISABLED*, que indica que esa opción se muestra, pero no se deja seleccionar.

**- Entrada de una lista de opciones**

Para permitir al usuario seleccionar varias opciones a la vez, se hace uso de **cajas** (*checkboxs*), que son pequeños cuadraditos que aparecen al principio de la opción, y que se pueden seleccionar o deseleccionar de manera independiente, pulsando con el ratón sobre ellos. El comando usado para este tipo de entrada es ***<INPUT TYPE=CHECKBOX NAME="nombre">***. Ejemplo:

```
<P>Deportes:</P>
<INPUT TYPE=CHECKBOX NAME="golf"> Golf
<INPUT TYPE=CHECKBOX NAME="tenis"> Tenis
<INPUT TYPE=CHECKBOX NAME="futbol"> Futbol
<INPUT TYPE=CHECKBOX NAME="natac"> Natacion
```

Cuando se envíe el resultado del formulario se pasa el valor ***NAME*** de la caja, y su valor será "***on***" si la opción está seleccionada. Opcionalmente se puede poner el atributo *CHECKED* si se desea que la opción aparezca seleccionada por defecto.

**- Botón de envío**

Permite enviar el formulario al programa (*ACTION*) del servidor que atiende el formulario. Se declara usando ***<INPUT TYPE=SUBMIT VALUE="texto">***, en donde *texto* es lo que aparecerá en el botón de pulsar. Ejemplo:

```
<INPUT TYPE=SUBMIT VALUE="Enviar solicitud">
```

**- Botón de limpiar formulario**

Permite devolver el formulario a sus valores originales por defecto. Se declara usando ***<INPUT TYPE=RESET VALUE="texto">***, en donde texto es lo que aparecerá en el botón de pulsar. Ejemplo:

```
<INPUT TYPE=RESET VALUE="Limpiar campos de pantalla">
```

Ejemplo de Formulario completo:

```
<FORM NAME="formulario" METHOD=POST ACTION="/cgi-bin/test">
<IMG SRC="imagenes/bolaroja.gif" HEIGHT=10 WIDTH=10 BORDER=0 HSPACE=2
SPACE=2 ALT="*">Nombre (puede dejarlo en blanco si lo desea):<BR>
<DD><INPUT NAME="nombre" TYPE=TEXT VALUE="" SIZE=50><BR>
<IMG SRC="imagenes/bolaroja.gif" HEIGHT=10 WIDTH=10 BORDER=0 HSPACE=2
SPACE=2 ALT="*">Sudirección de e-mail:<BR>
<DD><INPUT NAME="direccion" TYPE="text" VALUE="" SIZE=50><BR>
<IMG SRC="imagenes/bolaroja.gif" HEIGHT=10 WIDTH=10 BORDER=0 HSPACE=2
SPACE=2 ALT="*">Comentarios: <BR>
<DD><TEXTAREA NAME="comentarios" ROWS=10 COLS=50></TEXTAREA><BR>
<HR> <P>Si lo desea, puede darme su calificación:</P>
<TABLE BORDER=0 CELLSPACING=0 CELLPADDING=0>
<TR><TD>
<IMG SRC="imagenes/bolaroja.gif" HEIGHT=10 WIDTH=10 BORDER=0 HSPACE=2
SPACE=2 ALT="*">Contenidos:
<TD> <SELECT NAME="nota.cont">
<OPTION VALUE=" ">Seleccione una de las siguientes:
<OPTION VALUE="1">P&eacute;simo
<OPTION VALUE="2">Muy malo
<OPTION VALUE="3">Malo
<OPTION VALUE="4">Regular
<OPTION VALUE="5">Bueno
<OPTION VALUE="6">Excelente
</SELECT>
<TR><TD>
<IMG SRC="imagenes/bolaroja.gif" HEIGHT=10 WIDTH=10 BORDER=0 HSPACE=2
SPACE=2 ALT="*">Presentacion:
<TD> <SELECT NAME="nota.pres">
<OPTION VALUE=" ">Seleccione una de las siguientes:
<OPTION VALUE="1">P&eacute;simo
<OPTION VALUE="2">Muy malo
<OPTION VALUE="3">Malo
<OPTION VALUE="4">Regular
<OPTION VALUE="5">Bueno
<OPTION VALUE="6">Excelente
</SELECT> </TABLE>
<BR> <IMG SRC="imagenes/linea.gif" WIDTH="100%"><BR clear=all>
<BR> <INPUT TYPE=SUBMIT VALUE="Enviar comentarios">
<INPUT type=RESET Value="Limpiar formulario">
</FORM>
```

### 4.9. Marcos (Frames)

Una técnica ampliamente usada en los servicios multimedia existentes antes y después de la aparición del Web y del lenguaje HTML, era la de disponer de un documento con dos o más zonas, de manera que una de ellas mostrara información cambiante, y la otra información estática que servía de índice a la anterior.

Esta técnica permite una fácil navegación, ya que el usuario puede acceder de manera rápida a la información que desea.

Esta facilidad solo es soportada por el visualizador Netscape 2.0 o superior.

Antes de empezar a implementar una página Web que haga uso de FRAMES, se deben considerar los siguientes aspectos:

**. Sectorización**: consiste en definir la forma de dividir la página. Qué sectores contendrán los índices fijos, y qué sectores mostrarán la información variable. Lo más común es utilizar aproximadamente el tercio izquierdo de la página para el índice fijo y los dos tercios restantes para la información. Otra distribución típica es dejar la quinta parte superior de la página para los temas, o la información fija, o presentación de la entidad; y de lo que resta, el tercio izquierdo es para los índices fijos, y los dos tercios derechos para la información. La sectorización de la página se realiza con los comandos *FRAMESET* y *FRAME*.

**. Incidencia**: consiste en definir cuál será el sector (frame) que será afectado por la activación de cada enlace. Usualmente, los enlaces del índice muestran su resultado en el frame de la información, los del frame de los temas en el de los subtemas, y así sucesivamente. Los enlaces del marco de la información, suelen declararse dentro del mismo marco. También es posible que un enlace haga desaparecer la distribución de los marcos y se exprese en la página completa, o que cree un nueva ventana al ser activado.

El marco afectado por la activación de un enlace, se define mediante su atributo *TARGET*, con valor igual al *NAME* del marco que recibirá la página cargada. El TARGET también puede ser:

**_self**: valor por omisión, carga la página en el mismo FRAME del enlace.

**_parent**: carga la página en el FRAME del cual este FRAME es parte.

**_top**: carga la página a ventana completa.

**_blank**: inicia otra ventana.

Estos valores están reservados y ningún marco los puede llevar por nombre.

**- FRAMESET**

Un *FRAMESET* define un conjunto de marcos (*FRAME*) dispuestos uno sobre otro (*ROWS*), o bien, uno al lado de otro (*COLS*). Además, puede contener a otros *FRAMESET*, lo que posibilita múltiples sectorizaciones distintas. El ancho y la altura de cada marco se definen en el *FRAMESET*.

El comando *FRAMESET* acepta varias formas de definir la disposición de los marcos en su interior:

- *FRAMESET ROWS="20%,80%"* (porcentaje)
- *FRAMESET ROWS="20%,*"* ("*" indica todo el resto)
- *FRAMESET ROWS="30,*"* (pixeles)

Además permite utilizar varios atributos:

- *MARGINWIDTH*: indica el ancho de márgen del marco en pixels.
- *MARGINHEIGHT*: indica el alto de márgen del marco en pixels.
- *SCROLLING*: indica si este marco puede desplazarse (***yes/no/auto***). Con el valor *auto* es el visualizador quien decide si debe poder desplazarse o no (es el valor por defecto si no se pone).
- *NORESIZE*: indica al visualizador que no deje modificar el tamaño del marco al usuario.

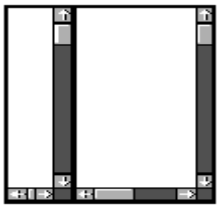
**- FRAME**

Cada comando *FRAME* tiene asociado un nombre (atributo *NAME*), que sirve para distinguir su zona de las otras, y una página de texto fuente (atributo *SRC*), que consiste en un **URL** apuntando hacia un documento HTML que será cargado en el marco al inicio.

Adicionalmente, se puede especificar el ancho y la altura del márgen, la presencia o ausencia de barras de desplazamiento, etc.

**- Ejemplo de división vertical**

```
<FRAMESET COLS="30%,70%">
<FRAME NAME="Indice" SRC="temas.html">
<FRAME NAME="Informacion" SRC="info.html">
</FRAMESET>
```

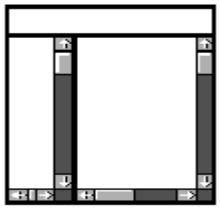


**- Ejemplo de división en dos mitades horizontales**

```
<FRAMESET ROWS="50%,50%">
<FRAME NAME="Superior" SRC="sup.html">
<FRAME NAME="Inferior" SRC="inf.html">
</FRAMESET>
```

**- Ejemplo de división mixta ,con marco superior sin barras de desplazamiento**

```
<FRAMESET ROWS="20%,80%">
<FRAME NAME="TITULO" SRC="titulo.html" SCROLL=NO>
<FRAMESET COLS="30%,70%">
<FRAME NAME="Indice" SRC="temas.html">
<FRAME NAME="Informacion" SRC="info.html">
</FRAMESET>
</FRAMESET>
```



**- Ejemplo completo**

**Archivo** index.html:

```
<HEAD>
<TITLE>Ejemplo de MARCOS</TITLE>
</HEAD>
<FRAMESET COLS="30%,70%">
<FRAME NAME="panel1" SRC="indice.html">
<FRAME NAME="panel2" SRC="tema1.html">
</FRAMESET>
```

**Archivo** indice.html:

```
<BODY>
<A HREF="tema1.html" TARGET="panel2">Tema 1</A>
<A HREF="tema2.html" TARGET="panel2">Tema 2</A>
<A HREF="tema3.html" TARGET="panel2">Tema 3</A>
</BODY>
```

**Archivo** tema1.html:

```
<BODY>
<P>Este es el contenido del primer tema ...</P>
</BODY>
```

**Archivo** tema2.html:

```
<BODY>
<P>... este es el contenido del segundo tema ...</P>
</BODY>
```

**Archivo** tema3.f.html:

```
<BODY>
<P>... y este es el contenido del tercer tema.</P>
</BODY>
```

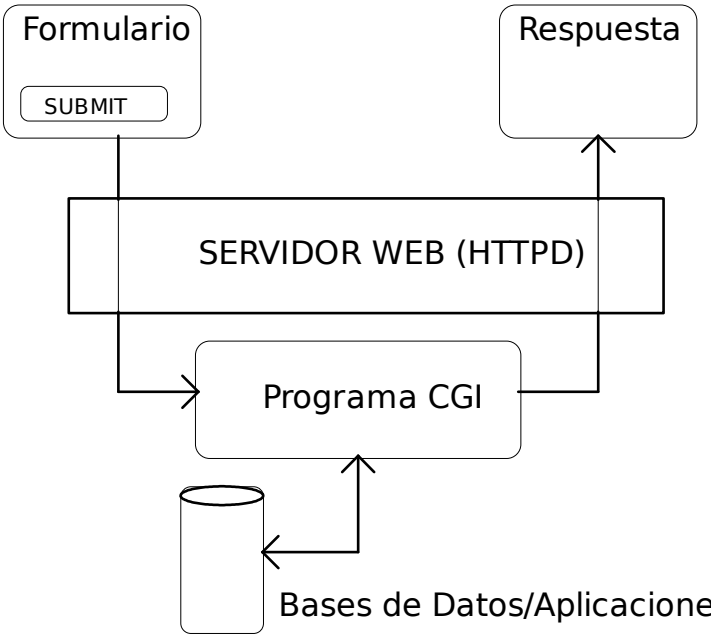
**- NOFRAMES**

Actualmente, sólo Netscape (2.0 o superior) reconoce la construcción de marcos. Para evitar problemas con los clientes que no los reconocen se debería incluir en el archivo, código suficiente para generar una página sin marcos. Para que este texto sea ignorado por el cliente que sí soporta marcos, se debe delimitar este texto mediante el comando ***<NOFRAMES>...</NOFRAMES>***.

La declaración de *FRAMES* se debe hacer antes de la aparición del comando *BODY* en el documento para que sea válida.

## 5. CGI

El siguiente texto es un documento introductorio y práctico para crear y usar programas ejecutables CGI, que son los que reciben los datos de los formularios. Un ejecutable recoge los campos de entrada de un formulario rellenos por un usuario, realiza una función y devuelve como resultado una página HTML al usuario. Su función suele ser consultar o hacer cambios en el disco del servidor, acceder a otros servicios, abrir una conexión a otro computador, etc.




Para crear un ejecutable CGI se puede usar cualquier lenguaje que pueda generar ejecutables (shell, Perl, C). Se necesita conocer alguno de estos lenguajes para realizar este tipo de programas.

El mecanismo de paso de información del servidor Web a la pasarela se puede hacer de dos formas:

**- Método GET**

El paso de campos se hace por medio de argumentos en la línea de comados. Un programa shell o Perl debe leer $1, $2, etc. Un programa C usa argv y argc.

**- Método POST**

El paso de campos se hace por medio de variables de entorno, es el que hay que utilizar cuando el formulario tiene muchos campos o campos con mucho contenido. Un programa shell o Perl tiene disponibles las variables. Un programa C hace utiliza getenv().

### Instalación de un ejecutable CGI

Esto dependerá casi por completo de la configuración de la que se disponga en el servidor Web. Por lo general, los ejecutables CGI tienen permiso de ejecución para todos, y se ubican en el directorio */usr/local/etc/httpd/cgi-bin*, o el equivalente en su máquina UNIX.

El directorio ~/cgi-bin de algún usuario (en estricto rigor se trata de su directorio public_html/cgi-bin o pub_www/cgi-bin).

Los programas y el directorio */usr/local/etc/httpd/cgi-bin* tendrán permisos de escritura, lectura y ejecución para todo el mundo.

### Un programa de prueba con Shell

Un procesador de formularios en *SHELL*, contiene un conjunto de instrucciones donde recoge las variables correspondientes al formulario. Se muestra aquí un ejemplo que recoge todas las variables y genera una página con ellas, para mostrársela al usuario.

```
#!/bin/sh
echo Content-type: text/plain
echo CGI/1.0 test script report:
echo argc is $#. argv is "$*".
echo
echo SERVER_NAME = $SERVER_NAME
echo GATEWAY_INTERFACE = $GATEWAY_INTERFACE
echo SERVER_PROTOCOL = $SERVER_PROTOCOL
echo SERVER_PORT = $SERVER_PORT
echo REQUEST_METHOD = $REQUEST_METHOD
echo HTTP_ACCEPT = "$HTTP_ACCEPT"
echo PATH_INFO = "$PATH_INFO"
echo PATH_TRANSLATED = "$PATH_TRANSLATED"
echo QUERY_STRING = "$QUERY_STRING"
echo REMOTE_HOST = $REMOTE_HOST
echo REMOTE_ADDR = $REMOTE_ADDR
echo REMOTE_USER = $REMOTE_USER
echo AUTH_TYPE = $AUTH_TYPE
echo CONTENT_TYPE = $CONTENT_TYPE
echo CONTENT_LENGTH = $CONTENT_LENGTH
```

### Un MAIL con PERL

Este ejemplo envía un mail al usuario especificado en *$destino*. Primero contiene un encabezado que deja todas las variables del formulario en la variable *FORM*, que es una cadena indexada, donde las claves son los nombres de las variables. Así, por ejemplo, *$FORM{'email'}* se referirá al valor del elemento que tenía *NAME=email* en el formulario. Este encabezado es típico de todos lo *CGI* escritos en *PERL* que utilizan el método *POST*.

```
#!/usr/local/bin/perl
$mailprog = '/usr/bin/mail';
$destino = 'pepe@empresa.es';
$respuesta = 'gracias.html';
#Inicio para que el cliente reconozca que es HiperTexto
print "Content-type: text/html\n\n";
system( "cat $respuesta" );
# Lee la entrada. Solo funciona con metodo POST.
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
# Separa la entrada POST en nombres-valores
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{ ($name, $value) = split(/=/, $pair);
  $value =~ tr/+/ /;
  $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
  $value =~ s/~!/ ~!/g;
  $FORM{$name} = $value; }
# Envia el MAIL a $destino
open (MAIL, "|$mailprog $destino") || die "No puedo abrir $mailprog!\n";
print MAIL "CURSO WEB - COMENTARIOS: \n\n";
print MAIL "Nombre: $FORM{'nombre'}";
print MAIL "\n-------------------------------------------------\n\n";
print MAIL "$FORM{'direccion'}";
print MAIL "\n-------------------------------------------------\n\n";
print MAIL "Comentarios :$FORM{'comentarios'}\n";
print MAIL "\n-------------------------------------------------\n\n";
print MAIL "Origen: $ENV{'REMOTE_HOST'} \n";
print MAIL "Direccion IP: $ENV{'REMOTE_ADDR'} \n";
close (MAIL);
```

## 6. SERVER SIDE INCLUDES

Es una facilidad que permite incluir programas en una página WEB (con formato SHTML), que se ejecutan cuando el usuario solicita esa página, y justo antes de que el servidor la envíe. Se utiliza el comando: ***<!--#programa [variables] -->***. Donde ***<!--#*** indica que viene un SSI. ***programa*** es el nombre del programa ejecutable dentro del servidor, que se ejecuta, ***[variables]*** son los valores pasados al programa ***y -->*** es el final del comando. Tal y como se observa el comando ***<!--*** es el comando de comentarios, por ello si el visualizador recibe esta línea tal cual, no la muestra.

El programa ejecutable puede ser realizado por el usuario o puede utilizar uno de los predefinidos como los siguientes:

- **echo**: realiza la visualización de una variable. Se utiliza así: ***<--#echo var="Variable de Entorno" -->***. Donde las variables pueden ser:

DOCUMENT_NAME: nombre completo del documento o página.
DOCUMENT_URI: nombre local referenciado del documento o página a la base del servidor.
QUERY_STRING_UNESCAPED: variable con los campos del formulario y las entradas del usuario.
QUERY_STRING: variable con los campos (separados) del formulario y las entradas del usuario.
DATE_LOCAL: fecha y hora local del servidor.
DATE_GMT: fecha y hora GMT del servidor.
LAST_MODIFIED: fecha y hora de la última modificación del documento o página.
REMOTE_ADDR: dirección IP del cliente.
SERVER_SOFTWARE: nombre del software HTTP del servidor.
SERVER_NAME: nombre del servidor.
GATEWAY_INTERFACE: nombre del CGI.
SERVER_PROTOCOL: nombre y versión del software HTTP del servidor.
SERVER_PORT: puerto IP del servidor HTTP.
REQUEST_METHOD: método del formulario (GET o PUT).
PATH_INFO: directorio virtual del documento o página.
PATH_TRANSLATED: directorio físico o virtual del documento o página.
SCRIPT_NAME: nombre virtual del programa que se ejecuta.
REMOTE_HOST: nombre del ordenador del cliente.
AUTH_TYPE: método de autentificación del cliente.
REMOTE_USER: nombre de usuario cliente.
REMOTE_IDENT: identificación (RFC931) de usuario cliente.
CONTENT_TYPE: tipo de contenido.
CONTENT_LENGTH: longitud del contenido.
HTTP_ACCEPT: tipos de MIME que acepta el visualizador cliente.
HTTP_USER_AGENT: nombre del software del visualizador cliente.
REFERER: nombre del URL solicitado por el cliente.
FROM: dirección de correo electrónico del usuario cliente.
FORWARDED: nombre del servidor PROXY.
ACCEPT_LANGUAGE: lista de lenguas humanas aceptadas por el cliente.

- **include**: permite incluir un fichero en la página. Se utiliza así: ***<--#include virtual="fichero" -->*** o así: ***<--#include file="fichero" -->***. En donde *fichero* es el nombre del fichero a incluir. Si se utiliza *virtual* el nombre del fichero es relativo al directorio base del servidor HTTP, y si se utiliza *file* el nombre del fichero es el que le corresponde en la máquina servidor. Es recursivo.

**- fsize**: presenta el tamaño de un fichero. Tiene el parámetro *virtual* y *file* igual que el anterior. Utilizándose y significando lo mismo respecto al fichero.

**- flastmod**: presenta la hora y fecha de la última modificación de un fichero. Tiene el parámetro *virtual* y *file* igual que el anterior. Utilizándose y significando lo mismo respecto al fichero.

**- exec**: provoca la ejecución de un programa ejecutable dentro del Servidor.
Ejemplo: ***<!--#cmd=programa.ejecutable [variable1 [variable2 [...]]] -->*** o también: ***<!--#cgi=programa.ejecutable [variable1 [variable2 [...]]] -->***

**- config**: permite seleccionar los valores de determinadas variables u opciones de salida del servidor HTTP. Estas variables u opciones son:

. Errmsg: variable que contiene el mensaje que el servidor HTTP muestra cuando existe algún error al ejecutar un SSI.

. Timefmt: es la variable que indica el tipo de fecha y hora usada por el servidor.

. Sizefmt: es la variable que indica el tipo de formato usado para mostrar el tamaño de los ficheros por el servidor.

. Cmdecho: es la variable que indica si se muestran los comandos SSI o no.

. Cmdprefix: es la variable que indica el prefijo (directorio) a anteponer al nombre de los programa a ejecutar mediante SSI.

. Cmdposfix: es la variable que indica el sufijo (extensión normalmente) a posponer al nombre de los programa a ejecutar mediante SSI.

. Onerror: indica a donde se debe saltar en caso de error en la ejecución de un SSI.

Ejemplo: ***<!--#config variable=xxxxxx -->***

**- odbc**: permite gestionar una base de datso odbc. Tiene los siguientes atributos:

. Debug: permite visualizar un mensaje que indica que se está en modo de depuración. En este modo la ejecución del SSI informa al cliente de lo que está haciendo.

. Connect: indica el nombre de la base de datos odbc, y el nombre del usuario con su contraseña, que va a operar sobre la base de datos (tendrá que estar autorizado).

. Statement: es la instrucción SQL a realizar sobre la base de datos.

. Format: formato de presentación de las líneas, resultado de ejecución de la sentencia. Este formato es similar al del lenguaje C, con la salvedad de que solo permite como formatos de salida los de tipo *%s*. Ejemplo: ***format="Nombre: %s. Desc.: %s"***.

Sintaxis: ***<!--#odbc [debug="....."] [connect=bdatos,usuario,contraseña] statement=sentencia [format="...."] -->***
Ejemplo: ***<!--#odbc statement="SELECT NOMBRE, DESCRIPCION FROM CURSO ORDER BY 1" -->***

**- email**: genera un correo electrónico cada vez que se solicita la página. Tiene los siguientes atributos:

. fromhost: atributo obligatorio que indica el ordenador origen (SMTP) del correo.
. tohost: atributo obligatorio que indica el ordenador destino (SMTP) del correo.
. fromaddress: atributo obligatorio que indica el usuario origen del correo.
. toaddress: atributo obligatorio que indica el usuario destino del correo.
. message: cuerpo del mensaje.
. subject: tema del correo.
. sender: dirección de correo del que origina el correo.
. replyto: es la dirección a donde se debe responder el mensaje.
. cc: indica las direcciones de otros destinatarios que recibirán copia del mensaje.
. inreplyto: define el campo homónimo del mensaje.
. id: es el campo de identificación del mensaje.

Ejemplo, supongamos que tenemos un formulario que tiene los campos de *nombre*, *direccion* y *comentarios* y queremos que cada vez que algún cliente envía el formulario nos llegue un mail con esta información:

```
<!--#email fromhost=empresa.es tohost="empresax.es"
fromaddress=&&direccion&& toaddress="pperez@empresa.es" subject="Correo
automatico pagina xxx"
message="Comentarios de &&nombre&&:
&&comentario&&."
```

**- if**: permite comprobar condiciones y realizar saltos.

Sintaxis: `<!--#if <operando1><operador><operando2><operacion>`

Donde `<operando1>` y `<operando2>` son los operandos (cadenas, o números enteros o punto flotante), `<operador>` es un operador ("==", "*!*=", "<", ">", "*!*>", "*!*<"), y `<operacion>` es la operación a realizar si se cumple la condición (*goto*, '*print*', '*error*', *break*, '*errorbreak*', '*printbreak*').

Existe un operando especial denominado hasstring que compara la existencia de una cadena dentro de otra.

Las operaciones son:

. 'goto': hace que cuando se cumple la condición se salte a una etiqueta del texto.
. 'print': hace que se muestre el texto que sigue al print.
. 'error': muestra un mensaje de error.
. 'break': se interrumpe en este punto la transmisión del documento.
. 'errorbreak': muestra un mensaje de error y se interrumpe en este punto la transmisión del documento.
. 'printbreak': muestra el texto que hay a continuación y se interrumpe en este punto la transmisión del documento.

Ejemplos:

```
<!--#if "&&HTTP_USER_AGENT&&" hasstring "Mosaic" goto mosaic -->
<!--#if "CampoFormulario" == "" printbreak "<P>Debe teclear algo en el
CampoFormulario." -->
```

**- goto**: provoca que se salte a la zona del documento donde está el nombre de la etiqueta. Esto provoca que no se continúe transmitiendo el documento a partir de esta línea, si no que continúe a partir de la etiqueta. Ejemplo: `<!--#goto="misalto" -->`

**- label**: asigna un nombre de etiqueta a esta zona del documento, adonde se pueden realizar saltos. Ejemplo: `<!--#label="misalto" -->`

**- break**: provoca la finalización de la transmisión del documento en este punto. Ejemplo: ***<!--#break -->***

### 6.1. Un contador SSI con Shell

Para realizar un contador utilizando un SSI se puede utilizar un programa escrito en *SHELL*. Este programa requiere un fichero donde guarda el número del contador que se va incrementando. El nombre de este fichero se recibe como parámetro, de tal forma que cada página puede tener su propio contador. El programa es el siguiente:

```
#!/bin/sh
#echo $1
#COUNTER='/usr/local/etc/httpd/index.cnt'
COUNTER=$1
if [ \! -w $COUNTER ]
then
        # no existe contador. Se inicia uno nuevo
        COUNT='1'
        echo Inicio de Contador
        echo $COUNT >$COUNTER
else
        # actualiza el contador
        COUNT=`cat $COUNTER`
        COUNT=`expr $COUNT + 1`
        echo $COUNT >$COUNTER
fi

echo Visita Num.: $COUNT
```

La forma de incluirlo en una página es:

```
<!--#exec cmd="contador.sh pagina.cnt" -->
```

## 7. JAVA

JAVA es un lenguaje de programación de propósito general con las cualidades siguientes:

- Orientado a objetos.
  - · se basa en clases y objetos
  - · incorpora abstracción y encapsulación de datos
  - · incorpora herencia
- Interpretado.
- Portable, independiente de la arquitectura. El compilador genera "byte codes" que son ejecutados por intérpretes en cada ordenador específico.
- Simple. Basado en C++, pero evita algunas de las difultades de este lenguaje.
- Robusto y seguro, con dos niveles de comprobación. Uno en tiempo de compilación y otro en tiempo de ejecución.
- Dinámico: las clases se enlazan en tiempo de ejecución.

Este lenguaje es similar a C++. Por ello es conveniente conocer C++ para poder comprender fácilmente este capítulo.

**Java solo es soportado por el visualizador Netscape 2.0 o superior y por Microsoft Explorer.**

El lenguaje **Java** permite que el visualizador realice algunas tareas al cargar una página. Entre estas tareas, puede estar, por ejemplo, realizar algunos cálculos simples, formatear un texto para que sea leído por distintas personas de manera distinta, proveer de un medio de configurar la visualización de una página, realizar comprobaciones de validación en un formulario antes de enviarlo, etc.

Existen dos tipos de programas JAVA:

- **Applet**, son programas escritos en Java, y compilados. Al visualizador no se le envía un texto, sino que se le envía un programa compilado (ByteCode).
- **Javascript**, son programas en formato texto que el visualizador interpreta.

### 7.1. Especificación del lenguaje JAVA

**- Sintaxis básica**

Es muy similar a la de C. Las instrucciones terminan con un punto y coma (;), se pueden agrupar usando llaves ({}), un doble-slash (//) indica que el resto de la línea es un comentario.Las asignaciones de variables se realizan usando el signo igual (=).

**- Tipos de datos**

- Enteros: byte, short, int, long (8, 16, 32 y 64 bits)
- Reales: float, double (32 y 64 bits)
- Caracteres: char (16 bits)
- Lógicos: boolean

No hay unsigned
Ejemplos:

```
var t2;
var resultado;
t1 = 2;
```

**- Operadores**

Los mismos que en C y C++ (+, -, *, /).

**- Arrays (cadenas)**

Igual que en C y C++

**- Strings**

Diferente que en C y C++, los strings no son arrays de caracteres, sino un tipo distinto:
String cadena = “Esto no es un array”;
Se pueden concatenar strings de manera muy simple usando el operador de suma (+).

**- Garbage Collector**

En JAVA no hay punteros, no hay que preocuparse por destruir objetos pues lo hace automáticamente un proceso en background.

**- Clases y Objetos**

La misma idea que en C++. Las clases son tipos especiales de datos. Y un objeto es cada una de las variables de un tipo (también se llaman instancias). Ejemplo de clase:

```
class Punto {
      private float x;
      private float y;
      public Punto() {
            x = 0.0;
            y = 0.0;  } }
```

Una clase tiene métodos y variables instancia, a éstas se accede a través de los métodos. Un tipo especial de método es el constructor.

La herencia se llama derivación en Java. Ejemplo de herencia:

```
class PuntoConColor extends Punto {
      private int color; }
```

**- Paso de Variables**

Los tipos de datos primitivos se pasan por **valor**.

Los tipos no-primitivos (arrays y objetos) se pasan por **referencia**.

**- Palabras reservadas**

Las palabras reservadas son: alert, abstract, boolean, byte, case, catch, char, class, const, default, do, double, extends, false, final, finally, float, goto, implements, import, instanceof, int, interface, long, native, null, package, private, protected, public, short, static, super, switch, synchronized, throw, throws, transient, try, void.

**- Características que se han eliminado de C++**

En JAVA no existen los siguientes elementos, que sí existen en C++:

. No hay preprocesador, por tanto se han eliminado los include, los typedefs y los ficheros de cabecera. En vez de éstos últimos tenemos interfaces del lenguaje con definiciones de clases y sus métodos. Esto hace que los programas sean independientes del contexto, puesto que no hay que conocer los ficheros de include para entender un programa. El efecto de los defines se consigue con variables const (como en C++)
. No hay estructuras ni uniones.
. No hay funciones.
. No hay herencia múltiple como en C++, evita problemas
. No hay goto, no se necesita.
. No hay sobrecarga de operadores como en C++.
. No hay casting automático, tienen que hacerse explícitamente.
. ¡No hay punteros!

### 7.2. Applet de JAVA

Para crear un Applet es necesario construir un programa (texto) utilizando el lenguaje JAVA. A continuación se compila, para generar otro fichero denominado ByteCode. Este es el programa que realmente se ejecuta (mediante un intérprete de Java), y que ve el usuario en su página. El ByteCode se ejecuta en el ordenador del cliente, no en el servidor.

Para incluir un Applet en una página se utiliza el siguiente comando:

```
<APPLET [CODEBASE = codebaseURL] CODE = appletFich [ALT = Texto] [NAME =
appletName] WIDTH = pixels HEIGHT = pixels [ALIGN = alineacion]>
[<PARAM NAME = appletAtributo1 VALUE = valor1>]
[<PARAM NAME = appletAtributo2 VALUE = valor2>]
. . .
</APPLET>
```

Donde los diferentes atributos significan lo siguiente:

. *CODEBASE = codebaseURL*: atributo opcional con el directorio base del Applet. Si no se especifica se utiliza el de la página.
. *CODE = appletFich*: atributo obligatorio con el nombre del fichero del Applet.
. *ALT = Texto*: texto alternativo que muestra el visualizador cuando no puede ejecutar el Applet.
. *NAME = appletName*: nombre del Applet.
. *WIDTH = pixels HEIGHT = pixels*: atributos obligatorios con el ancho y alto del área inicial de presentación.
. *ALIGN = alineacion*: tipo de alineación (*left*, *right*, *top*, *texttop*, *middle*, *absmiddle*, *baseline*, *bottom*, *absbottom*).
. <*PARAM NAME = appletAtribute1 VALUE = valor1*>: parámetro a pasar al Applet. Este accede al parámetro mediante el método getParameter().

### 7.3. JavaScript

El lenguaje **JavaScript** es similar a **Java**, pero no está compilado. Permite que el visualizador realice algunas tareas al cargar una página, sin necesidad de tener el compilador de **Java** en el Servidor. El programa JavaScript va insertado en la página HTML de forma visible (fuente).

**Esta facilidad solo es soportada por el visualizador Netscape 2.0 o superior.**

**JavaScript** recibe información a través de eventos y propiedades de objetos, y la entrega mediante propiedades de objetos y métodos.

**- Como incluir JavaScript**

Existen principalmente tres lugares de una página donde puede aparecer un trozo de lenguaje **JavaScript**:

1. Usando la directiva: ***<SCRIPT LANGUAGE="JavaScript">...</SCRIPT>***.

Por ejemplo:

```
<SCRIPT LANGUAGE="JavaScript">
<!--document.write( 'Esto es una prueba' );// -->
<SCRIPT>
```

2. Utilizado como respuesta a algún evento. Por ejemplo:

```
<INPUT TYPE="checkbox" onCLICK="window.status='Usted acaba de hacer click.'; return true;">
```

3. Incluido en el atributo *HREF* del elemento ***<A>***. Por ejemplo:

```
<A HREF="javascript:window.history.back()">Volver</A>
```

Las dos primeras son más usadas que la última.

**- Variables**

La declaración de una variable se puede hacer de dos formas: la primera como variable global, si se hace fuera de una función, y la segunda como local, declarándola dentro una función. Se puede declarar una variable sin indicar explícitamente su tipo, usando la instrucción:

```
var Nombre;
```

Más adelante en el programa el mismo intérprete le asigna el tipo apropiado.

Para declarar cadenas se utiliza lo siguiente:

```
var cadena = new Array();
cadena[0] = 2;
cadena[1] = "Elemento 1";
cadena[2] = "Elemento 2";
```

Dado que no está completamente implementada la forma de definir el número de elementos, se puede guardar la longitud de la cadena en el elemento cero.

Para crear objetos:

```
var Autor = new Object();
Autor.nombre = "Carlos Castillo";
Autor.apodo = "ChaTo";
Autor.edad = 19;
```

También se pueden crear cadenas asociativas, usando el mismo tipo de constructor:

```
var TablaColor = new Object();
TablaColor["rojo"] = "FF0000";
TablaColor["azul"] = "0000FF";
TablaColor["verde"] = "00FF00";
```

**- Control lógico**

Existen las instrucciones: if ... else, for, while ... break ... continue, with, function ... return, que funcionan de manera idéntica a C, y la instruccion for ... in. Ejemplos:

```
// Suponiendo que la variable myvar existe, y que hay un formulario
// llamado "myform" que tiene un input de tipo text llamado "texto1"
if ( myvar == 0 ) {
document.myform.texto1.value = 'Error'; }
else {
document.myform.submit(); }
// Aqui habra un error si el formulario "myform" tiene menos de 31 elementos.
// los elementos en general en JavaScript se enumeran comenzando por el
// elemento cero ( las opciones de un select, o de un radiobutton, o los
// elementos de un documento o los elementos de un formulario, o los
// formularios de un documento )
for ( myvar = 0; myvar <= 30; myvar++ )
document.myform.elements[myvar].value = '';
// Un ciclo while tipico que no hace nada excepto hacer position = 30
var position;
position = 0;
while ( position < 30 )
position++;
// Una vez especificado with, el resto de las variables utilizadas
// comienzan con el argumento de with
// se asume que existen dos formularios en el documento, el segundo
// formulario puede no tener nombre, por lo que se referencia por
// su numero dentro del arreglo especial 'form'
with ( document.form[1] ) {
nombre.value = "Introduzca aqui su nombre";
edad.value = "Puede dejarme en blanco si lo desea"; }
// for ... in ejecuta una accion con cada elemento de un arreglo
for contador in Arreglo {
write( Arreglo[contador] ); }
```

**- Funciones de Depuracion**

La instrucción alert es muy útil en el momento de programar y depurar código **JavaScript**. Su uso es muy simple: alert( "*mensaje*" );

**- Funciones**

Para declarar una función se usa la instrucción function.

```
function NombreFuncion (parametro1, ..., parametro N) {
...
return valor;
}
```

No existe un cuerpo principal del programa (main), puesto que todo lo que no esté dentro de una función es ejecutado mientras se va cargando la página, conforme va apareciendo. Cuando se declara una función, tampoco es necesario indicar que tipo de valor retornará, si es que retorna alguno. Una función debe ser declarada antes de usarse.

**- Depuración**

La instrucción alert que es muy útil al momento de programar y depurar código **JavaScript**. Su uso es, por otra parte, muy simple: alert( "*mensaje*" );

**- Eventos**

Un evento debe asociarse a un elemento HTML, no a un código **JavaScript**. Generalmente se usan para invocar a funciones que realizan alguna operación con el mismo elemento que las invoca, por eso, se define la palabra reservada "this" para referirse al objeto cuyo evento invocó a la funcion. El ejemplo típico es sumar dos números:

```
<SCRIPT LANGUAGE="JavaScript">
<-- // esto sirve para esconder el script de browsers antiguos
var valor1 = 0;
var valor2 = 0;
function Set1 ( entrada ) {
  valor1 = parseInt( entrada.value ); }
function Set2 ( entrada ) {
  valor2 = parseInt( entrada.value ); }
function totalizar ( boton ) {
  document.myform.total.value = valor1 + valor2; }
// -->
</SCRIPT>
<FORM NAME="myform">
Numero 1: <INPUT TYPE="text" NAME="numero1" onChange="Set1( this )">
Numero 2: <INPUT TYPE="text" NAME="numero2" onChange="Set2( this )">
<INPUT TYPE="button" VALUE="Sumar!" onClick="totalizar( this )">
<INPUT TYPE="text" NAME="total">
</FORM>
```

Los eventos definidos son los siguientes:

. **onBlur**: se produce este evento cuando se pierde el cursor en el campo de un formulario del tipo *text*, *select* o *textarea*. Ejemplo:

```
<INPUT TYPE="text" onBlur="comprueba_valor(this.value )">
```

. **onChange**: al cambiar o perder el cursor en text, select, textarea. Ejemplo:

```
<INPUT TYPE="text" onChange="Set1( this.value )">
```

. **onClick**: al pulsar el ratón sobre un botón tipo *button*, *checkbox*, *link*, *radio*. Ejemplo:

```
<INPUT TYPE="button" VALUE="Enviar 1" onClick="document.myform.oculto1 =
1; document.myform.submit();">
```

. **onFocus**: al ganar el cursor en *text*, *select*, *textarea*. Ejemplo:

```
<TEXTAREA onFocus="this.value = ''">Esto se borrara</TEXTAREA>
```

. **onLoad**: al cargarse una página *body*. Ejemplo:

```
<BODY onLoad="preparar_formulario( document.myform )">
```

. **onUnload**: al descargarse una página *body*. Ejemplo:

```
<BODY onUnload="document.ocultaform.submit()">
```

. **onMouseOver**: al pasar el ratón por encima de un enlace. Ejemplo:

```
<A HREF="../..." onMouseOver="window.status = 'Volver atras'">
```

. **onSelect**: al seleccionar texto en *text*, *textarea*. Ejemplo:

```
<INPUT TYPE="text" onSelect="window.status='Presione espacio ahora para
borrar el texto seleccionado'">
```

. **onSubmit**: al enviar un formulario. Ejemplo:

```
<FORM onSubmit="window.defaultStatus = 'Espere un momento mientras se
procesa el formulario ...'">
```

**- Objetos**

Los siguientes son algunos de los objetos con los que los scripts pueden interactuar:

- Objetos Globales:
  . 'document':el documento cargado.
  . 'window': la ventana activa.
  . 'form': un formulario del documento, identificado por su atributo NAME además de todos sus elementos, identificados también por sus nombres.
  . 'history': la historia de la ventana o de un frame.

- Objetos especiales
  . 'Date': un objeto genérico conteniendo una fecha.
  . 'Math': una biblioteca incorporada de funciones matemáticas y constantes.

Ejemplos:

```
window.history.go(0) // recarga la página actual (método)
document.referrer    // contiene el URL de la página anterior (no puede
                        modificarse)
window.status        // la barra de estado (sí puede modificarse)
window.history.back()// se retrocede (método equivalente a
                        window.history.go(1)
var pi = Math.pi     // biblioteca de funciones matemáticas
window.defaultStatus // la barra de estado
document.form1.submit() // envía el formulario llamado form1
window.location = 'index.html' // actúa con la ventana.
```

**- Ejemplo**

## 8. Herramientas y Utilidades

### 8.1. Editores HTML

En la actualidad existen muchos editores de páginas, muchos de ellos son software *shareware*. Los más famoso son *HotDog* y *HotMetal*. Aunque Netscape en su última versión 3.01 incorpora a su visualizador un potente editor integrado ya con el visualizador, lo cual es una notable ventaja. También los procesadores de texto como *Word* se han modernizado e incorporan esta nueva especificación de hipertexto. Aunque, conociendo el lenguaje HTML, cualquier editor (desde el NotePad de Windows) es suficiente para crear páginas.

En realidad, lo más importante son los procesadores de imagen: *Adobe PhotoShop*, *Corel*, *Corel Xara*, *PaintShop Pro*, filtros de *Kai*, *Gif Animator*, *Shockwave*, *Photo Impact*, *GIF Construction*, *GIFWeb*. Y para crear mapas sensibles *MapThis*.

### 8.2. Animaciones con GIF

El formato GIF89a, permite tener varias imágenes en un solo archivo GIF, junto con información para mostrar estas imágenes secuencialmente como una animación, con distintos retardos entre las imágenes o la posibilidad de que la animación se repita una y otra vez (loop).

Las animaciones GIF no requieren de un programa, ni de una aplicación (como ocurre con los ficheros de video o las animaciones quicktime), ocupan muy poco espacio, y se puede definir la posición de una imágen dentro de la animación, ahorrando así espacio en el archivo. Por ejemplo, si la animación es una persona caminando, basta grabar el fondo una sola vez en el archivo, y luego la persona, indicando la posición en donde esta segunda imágen se posicionará. En los otros sistemas de video, hay que grabar la imágen de todo el cuadro.

Sin embargo, el formato no es reconocido por todos los browser. Así un visualizador que no reconozca la animación, mostrará el último cuadro como una imágen fija. Además cuando se especifica animación indefinida (loop), el visualizador puede ponerse muy lento y es incómodo para la persona que ve la imágen (el hecho de que se cargue la imágen una y otra vez).

Para crear las imágenes animadas GIF, primero se han de crear una serie de cuadros usando cualquier programa de dibujo (por ejemplo, **PaintShop Pro**, **Corel**, **Adobe PhotoShop**), y luego se graba en formato GIF, después se crea la animación mediante los cuadros, con el programa **GIF Construction** (para Windows). Para crear transparencias en imagenes GIF se utiliza el programa **GIFWeb** (para Windows).

### 8.3. Animación de Páginas

Para realizar animación de páginas existen varios métodos. Uno es el denominado ***Server Push*** que consiste en que el servidor envía al visualizador del cliente una misma página o imagen cada cierto tiempo. Produciendo el efecto de animación. Otro consiste en utilizar un programa ***Aplet Java*** o ***JavaScript*** que realice la animación en el visualizador del cliente. Otro es el ***Client Pull*** que hace que el visualizador solicite la página cada cierto tiempo. El siguiente ejemplo se refresca cada quince segundos (no es buen método):

```
<HTML>
<META HTTP-EQUIV="Refresh" CONTENT=15>
<TITLE>Documento Uno</TITLE>
<H1>Este es el primer Documento</H1>
......
```

Para interrumpir el refresco se debe presionar uno de los botones de parar o atrás del visualizador o un enlace de la página. Si se desea cargar otra página se debe poner en el comando *META* la referencia completa de la otra página a cargar. Por ejemplo:

```
<META HTTP-EQUIV="Refresh" CONTENT=15 URL=http://ord.es/segunda.html">
```

De esta forma se puede hacer un recorrido automático por varias páginas cada cierto tiempo, con texto, imágenes, audio, video, etc.

### 8.4. Contadores de Páginas

Son programas que al ser puestos (ejecutados) en páginas indican el número de visitas que se ha realizado a una página. En el capítulo de SSI hay un ejemplo.

### 8.5. Propagadores de Páginas en Internet

Para dar a conocer nuestra página al resto del mundo Internet, es necesario rellenar un cuestionario en cada **buscador** (Yahoo, Lycos, AllInOne, Olé, ...). Para evitar este engorroso trabajo existen otros servidores denominados **Propagadores** que realizan esta labor por nosotros; de tal forma que rellenando un cuestionario en uno de estos propagadores, éste se encargará de realizar lo mismo en varios buscadores. Uno de estos propagadores es **Submit-It** y otro es **Atajos**, éste último en español.

## 9. GUIA DE ESTILO

Una página WEB es mucho más que un documento mecanografiado con imágenes, es un documento hipermedia, es una nueva dimensión de presentación de información que está naciendo.

Con el surgimiento de los servicios hipermedia, los recursos de este medio son un campo inexplorado en dos facetas. En primer lugar, en cuanto a los medios de transferencia de los documentos, los lenguajes, los protocolos, etc. En segundo lugar, en cuanto a la mejor manera de presentar la información en el hipertexto, la manera más clara, la que minimice el tiempo de búsqueda del usuario, la que haga más agradable su lectura, etc.

Existen múltiples normas acerca de cómo debería ser un buen documento HTML, y a pesar de lo interesante que son algunas de estas ideas, no hay nada que pueda reemplazar el buen juicio y la búsqueda de un estilo particular de hacer documentos. Así, el problema del estilo está inmerso en la perspectiva personal del creador. A pesar de todo, se mencionan un conjunto de reglas que es aconsejable respetar.

**- ¿Qué información ofrecer?**

¿ Otra lista de enlaces? **!POR SUPUESTO QUE NO!**. Todo el mundo conoce Yahoo, AllInOne y Lycos. Mejor tener solo los enlaces del tema que tratamos.
A veces hay excesiva información redundante o poco interesante, que lo que provoca es pérdida de tiempo y enfado a los usuarios que leen nuestras páginas. Es importante saber acerca de lo que se escribe, y que esto tenga sentido.

El intercambio de opiniones con personas de otras disciplinas puede ser muy enriquecedor, pues aportan una visión distinta de la que tienen los informáticos.

**- Sacar partido al hipertexto**

Las referencias cruzadas permiten abordar un mismo tema en distintos niveles de profundidad. Se puede permitir al usuario pasar de largo información técnica o conceptos avanzados de un tema, estableciendo enlaces a textos más extensos. En general, podemos disponer de una página corta, con un resumen de la información, y que lleve a páginas que contengan detalles más concretos.

Muchos enlaces también pueden llevar a confusión, sobre todo si no es posible inferir del contexto su contenido. En este punto sólo queda el criterio del autor.

**- Usar las capacidades multimedia**

Una imágen vale por mil palabras. Una animación contiene varias imágenes. La conclusión es obvia (pero no absoluta) en ocasiones una animacion se transforma en algo muy confuso).

**- Un medio de realimentación**

El elemento <ADDRESS>, el e-mail del autor, un pequeño ícono al final de la página, etc, permiten mantener una página HTML actualizada.

**- Tiempo de búsqueda. Indices y más índices**

Una clasificación adecuada de la información de nuestro servidor puede posibilitar expandir su uso a distintos públicos. En general, la página principal (homepage) de nuestro

servidor debería poder presentar en una sola página física toda la información disponible (muy abreviada y esquemática).

**- La velocidad de transferencia**

Una página con muchos Kb de imágenes aburre al más pertinaz buscador, sobre todo si accede por línea telefónica convencional a Internet. Una página sin imágenes tampoco es lo mejor. Un máximo de 10 a 15 Kb de gráficos por página es una buena norma, por lo menos para las primeras páginas que deba consultar el lector en su búsqueda.

**- Revisar el contenido periódicamente**

Una página con enlaces anticuados, o con servicios que no funcionan es algo simplemente *desalentador*. Una revisión periódica ahorra al usuario muchas molestias y permite mantener el interés de nuestras páginas WEB. Además, es importante indicar la **fecha de la última modificación** y las adiciones hechas. Esto permite entre otras cosas que el usuario se cerciore de la seriedad del autor. Si una página aparece con un signo "en construcción" y su última revisión es del año anterior, será un buen indicio de que es mejor buscar en otra parte.

**- Formato lógico mejor que formato físico**

Distintos usuarios pueden desear ver la información de distintas maneras, por ejemplo, unos querrán ver el texto resaltado en color azul y las citas en verde, o cualquier otra cosa por el estilo. Este es un concepto clave en HTML. Perfectamente en los inicios se podía haber utilizado otro lenguaje con más comandos como cambiar de tamaño, especificar posición, etc. Pero HTML no es un lenguaje para transportar textos. Es un lenguaje más bien para ordenar las ideas y posibilitar búsquedas rápidas. Antes de elegir el marcador conveniente, es bueno preguntarse: ¿qué quiero representar?.

*<EM>énfasis </EM>*
*<CITE>citas de texto</CITE>*
*<VAR>valores de variables</VAR>*
***<KBD>teclas</KBD>***
***<CODE>código fuente</CODE>***
***<LISTING>código fuente listado</LISTING>***
***<SAMP>muestras</SAMP>***
**<STRONG>Gran énfasis</STRONG>**

Los encabezados son textos resaltados, no son un mecanismo de selección de tipo de letra. No hay necesidad de transformar un montón de texto en encabezado sólo para que se vea resaltado. Esto no será aceptado por todos, y puede llegar a verse realmente feo si la persona que ve el documento no tiene la misma percepción que la persona que lo creó.

**- Páginas Actualizadas**

Mantenerse al día en cuanto a las nuevas adiciones de comandos, para que nuestros documentos puedan ser leídos por clientes hipertexto nuevos.

**- Buen uso de comandos HTML**

No es bueno separar los elementos de formato con comandos estructurales.
Por ejemplo ***<I>...<P>...</I>*** no es correcto.
Sin embargo ***<I>...</I><P><I>...</I>*** si lo es.

Tampoco se deben incluir comandos de formato dentro de otros comandos del mismo tipo o dentro de enlaces, o dentro de la cabecera de la página HTML.
Por ejemplo, **no se debería poner** un párrafo dentro de un encabezamiento.

```
<H2>...<P>...<P>...</H2>
```

Ni tampoco comandos HTML dentro de las referencias.

```
<A><H1>...</H1></A>
```

Ni texto visible dentro del encabezado.

```
<HEAD>
<TITLE>...</TITLE>
<H1>...</H1>
</HEAD>
```

**- Posición de enlaces**

La elección del lugar para poner enlaces es crucial para la presentación del hipertexto.
Por ejemplo:

```
El <A HREF="http://www.empresa.es/">Empresa XXX</A> fue fundada en 1952.
```

Es mejor que:

```
La Empresa XXX fue fundada en 1952. <A HREF="http://www.empresa.es/”>pulse
aqui para ver informacion acerca de la Empresa XXX </A>.
```

**- Propiedad Intelectual y reconocimiento**

No utilizar gráficos de otros sin autorización. Existen millones de imágenes libres de derechos de autor disponibles. Lo mismo ocurre con respecto al texto.

Además, es bueno proveer de enlaces a páginas que traten sobre el mismo tema o temas similares a la página que estamos escribiendo.