**T1. Introducción a los Lenguajes de Marcas.**

1. **Conceptos y ventajas.**

Una **marca históricamente** es una anotación que intenta señalar a un tipógrafo cómo debe imprimirse o prepararse un texto determinado.

**Marca en la actualidad** es toda clase de códigos insertados en textos electrónicos que determinan el formato, el modo de impresión o cualquier otro proceso.

Un **lenguaje de marcado** en los documentos electrónicos es un tipo de lenguaje que combina texto con información extra acerca del texto. Esa información extra se entremezcla con el texto primario.

El marcado define una serie de códigos llamados **etiquetas**:

<etiqueta>Texto sobre el que se actúa</etiqueta>

Definen la estructura del documento, es decir, los elementos con que se organiza dicho documento. Y además el formato en el que tienen que aparecer, refiriéndose a la presentación (tipo de fuente, negrita, cursiva…).

1. **SGML. El origen.**

IBM intenta resolver el problema de intercambio de documentos entre plataformas creando el **GML (Generalized Markup Language).**

GML independiza el documento del dispositivo que lo va a utilizar, usando marcas genéricas. Incorpora marcas descriptivas para la estructura que permiten distinguir el texto de las listas, las tablas, etc. El mismo documento puede ser entonces utilizados por varios dispositivos.

ISO convierte en 1986 GML en **SGML, Standard Generalized Markup Language**, el cual no es estrictamente un lenguaje sino un metalenguaje, un conjunto de normas que permiten crear otros lenguajes de marcas. Esto se hace definiendo un vocabulario o conjunto de elementos a utilizar y una gramática o conjunto de reglas que rigen el uso de los elementos y sus atributos.

1. **Clasificación de los Lenguajes de Marcas.**

Los lenguajes de marcas se clasifican en 3 tipos:

* **De presentación**. Indican el formato del texto o tipografía, sin especificar su estructura. Los procesadores de texto y en general las aplicaciones de edición profesional utilizan este tipo de marcado. (nroff, troff, RF, TeX, DocBook).
* **Descriptivo**. Indican las diferentes partes en las que se estructura el documento, pero sin especificar cómo deben representarse ni en qué orden. XML es un metalenguaje diseñado para generar marcado descriptivo.

Estos lenguajes crean documentos con estructura en árbol que almacenan información. Algunos ejemplos son YAML, EBML, RDF, XFML…

* **Híbrido**. Son lenguajes que contienen marcas de los dos tipos anteriores indistintamente.
* **Según la funcionalidad**, tenemos otra clasificación de lenguajes de marcas:
  1. Documentación electrónica: RTF, TeX, EBML, YAML.
  2. Tecnologías de Internet: HTML, XHTML, WML, XAML, RSS, Atom, WSDL, SOAP, XMPP.
  3. Propósito específico: MathML, SSML, MusicXML, SVG,VML, SMIL, XLL, XSLT.

1. **XML. Características.**

XML (**eXtensible Markup Language**), es una adaptación y simplificación de SGML, que permite definir lenguajes específicos. XML no es un lenguaje en particular, es una manera de definir lenguajes para diferentes necesidades. Sus **características básicas son**:

* **Extensible**, se pueden definir nuevas etiquetas.
* **Versátil**, separa contenido, estructura y presentación.
* **Estructurado**, se pueden modelar datos a cualquier nivel de complejidad.
* **Validable**, cada documento se puede validar frente a un DTD/Schema.
* **Abierto**, independiente de empresas, SO, lenguajes de programación o IDE.
* **Sencillo**, fácil de aprender y de usar.

**T2. HTML y XHTML. Lenguajes para la Web.**

1. **Evolución histórica.**

HTML **(HyperText Markup Language)**, aparece a principios de los 90 cuando el investigador Tim Berners Lee, lo desarrolla para potenciar la colaboración entre físicos e investigadores.

El **objetivo principal** era enlazar los documentos mediante links, de forma que al hacer clic sobre algún texto, se abriera otro documento relacionado con la información seleccionada.

1. **Estructura del documento HTML.**

Un documento HTML consta de las siguientes piezas:

* 1. Identificación SGML
  2. Una etiqueta <HTML>
  3. Cabecera (iniciada por la etiqueta <HEAD> y cerrada por </HEAD>)
  4. Cuerpo del documento (iniciada por la etiqueta BODY y cerrada por </BODY>)
  5. Una etiqueta de fin de documento </HTML>

<!DOCTYPE html>

<html>

<head>

<title>Título de la página</title>

</head>

<body>

Contenido del cuerpo

</body>

</html>

1. **Contenido de la cabecera <head>.**

Es un conjunto sin orden con información acerca del documento. Se identifica con la etiqueta <head> y finaliza por tanto con </head> En su ámbito se pueden emplear diferentes elementos referenciados por sus etiquetas, los más relevantes son:

1. **<title>.**

<title> [cadena de caracteres] </title>

Da título al documento, en la mayoría de los navegadores se visualiza en la barra de titulo.

1. **<base>.**

base href=“Url”>

Indica la localización de los ficheros, gráficos, sonidos, etc. a los que se hace referencia en nuestra página web. Si no se incluye esta directiva, el navegador entiende que los elementos se encuentran en el mismo lugar que nuestra página.

<base href=“http://127.0.0.1/images/”>

1. **<meta>.**

<meta [http-equiv=valor1] [name=valor2] content=valor3>

Los meta-tags sirven para dar información a los robots de los buscadores. Los metas name, dan la información a los robots, indicándoles algunas preferencias nuestras y del propio documento como el idioma en el que está escrito, el tipo de distribución.

Ejemplos:

<meta name=“Language” content=“ES”>

<meta name=“Keywords” CONTENT=“tutorial de html”>

<meta name=“Description” CONTENT=“curso de html”>

<meta name=“Distribution” CONTENT=“Global”>

<meta name=“Robots” CONTENT=“All”>

Forzar a que la página activa se cargue cada cierto tiempo (indicado en segundos mediante el atributo CONTENT).

<head>

<meta http-equiv=“refresh” content=“10”>

</head>

1. **<script>.**

Se incluye en la cabecera también todo el código Javascript, bien haciendo referencia a archivos .js con el atributo src de la etiqueta <script> o bien escribiendo el código Javascript dentro de las etiquetas <script>.

Ejemplo:

<script language=“javascript” type=“text/javascript”>

alert(“hola mundo”);

</script>

1. **<style>.**

Contiene los estilos CSS (más adelante veremos qué son) que se aplicarán a la página. Bien haciendo referencia a ellos con la etiqueta <link>, bien importándolos con la sentencia @import dentro de una etiqueta <style> o bien definiéndolos directamente dentro de una etiqueta <style>.

Ejemplo:

<head>

<link href=“especial.css” rel=“stylesheet” type=“text/css”>

Ejemplo: Una hoja de estilo puede incrustarse en un documento con el elemento STYLE.

Ejemplo: cambiar el color de la letra del cuerpo a rojo.

<style type=“text/css” media=screen>

body { color: red }

</style>

1. **Contenido del cuerpo <body>.**

Es el contenedor de la información propia del documento.

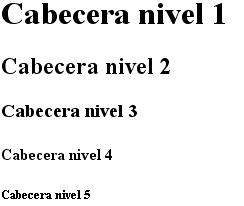
Se identifica con la etiqueta <body> y finaliza por tanto con la etiqueta </body>.

Los elementos del cuerpo del documento son los siguientes:

1. **Manejo del texto:**

* **Cabeceras. <h1|h2|h3|h4|h5|h6>.**

Existen 6 niveles de cabeceras:



<h1>cabecera de nivel 1</h1>

<h2>cabecera de nivel 2</h2>

<h3>cabecera de nivel 3</h3>

<h4>cabecera de nivel 4</h4>

<h5>cabecera de nivel 5</h5>

<h6>cabecera de nivel 6</h6>

* **Párrafo <p>.**

Conjunto de texto que empieza y acaba con un salto de línea, la etiqueta de finalización no es obligatoria. Admite el parámetro ALIGN=(left|center|right).

* **Salto de línea <br>.**

Fuerza que se parta una línea de texto independientemente del formato en que se este trabajando.

Extensiones de la etiqueta <br>:

<br clear=left> busca el primer margen libre (clear) a la izquierda.

<br clear=right> busca el primer margen libre a la derecha.

<br clear=all> busca el primer margen libre a ambos lados.

Su uso esta indicado para romper la secuencia de texto cuando se sitúa este alrededor de una imagen

* **Línea horizontal <hr>.**

<hr width= align= size=>

Dibuja una línea horizontal.

WIDTH= Tamaño de la largura horizontal de la línea

ALIGN= Alineación en la pagina: LEFT, RIGTH

SIZE= Grosor de la línea.

Ejemplo:

<hr width= “60%”>

* **Etiquetas de resaltado de de texto.**

|  |  |
| --- | --- |
| CODIGO | SALIDA |
| Texto Normal | Texto Normal |
| <CITE> ejemplo: Cita </cite> | *Cita* |
| <CODE> ejemplo: Ejemplo de código </code> | Ejemplo de código |
| <EM> ejemplo: Énfasis </em> | *Énfasis* |
| <KBD> ejemplo: Entrada de teclado </kbd> | Muestra |
| <SAMP> ejemplo: Muestra </samp> | **Énfasis fuerte** |
| <SAMP> ejemplo: Muestra </samp> | *Variable* |
| <STRONG> ejemplo: Énfasis fuerte </strong> | **Énfasis fuerte** |
| <VAR> ejemplo: Variable </var> | *Variable* |
| <SUP> ejemplo: Superíndice </sup> | Superíndice |
| <SUB> ejemplo: Subíndice </sub> | Subíndice |
| <B> ejemplo: Letra en negrita | **Letra en negrita** |
| <I> ejemplo: Letra inclinada </i> | ***Letra inclinada*** |
| <TT> ejemplo: Letra maquina de escribir </TT> | **Letra de maquina de escribir** |
| <U> ejemplo: Letra subrayada </u> | **Letra subrayada** |
| <BIG> ejemplo: Aumenta tamaño de fuente </big> | **Aumenta tamaño de fuente** |
| <SMALL> ejemplo: Disminuye tamaño de la fuente | **ejemplo: Disminuye el tamaño de la fuente** |

* **Fuente <font> (desaprobado>.**

<font size= color= face=>

Propiedades del tipo de letra.

* + Size: Tamaño (de 1 a 7, siendo 3 el tamaño por defecto) por asignación (SIZE=4) o con desplazamiento relativo al tamaño base (SIZE=+1).
  + Color: Color de la fuente en formato RGB.
  + Face: Nombre de la fuente a utilizar, es admitido por algunos navegadores, pero no esta incluido en la norma HTML 3.2.

Texto normal

<font size=+1 color=“#008000”>texto</font>

Salida

Texto normal texto

* **Texto con formato previo <prev>.**

Conjunto de texto que se muestra como se introdujo en el formato original.

HTML elimina cualquier espacio en blanco adicional que se inserte pero nos encontramos con una excepción a esta regla cuando utilizamos las tags <pre>.....</pre>. Sin embargo esta tag convertirá el texto afectado a fuente monoespaciada (posiblemente Courier).

Esta tag se usaba para hacer tablas en versiones anteriores del lenguaje HTML ahora su utilidad puede reducirse a convertir a HTML, rápida y fácilmente, archivos de correo electrónico entre otros.

1. **Listas.**

Existen tres tipos de listas:

* **Listas de elementos.**
* *Listas de elementos no ordenados <ul>.*

Lista de textos, cada entrada comienza por un carácter de señalización.

<ul>

<li> texto

<li> texto

</ul> ....

Salida:

* + texto
  + texto
* *Listas de elementos ordenados <ol>.*

Cada elemento de la lista comienza por un numero, igual a su posición en la lista.

<ol>

<li> texto

<li> texto

</ol>

Salida:

* + - 1. texto
      2. texto
* **Listas anidadas.**

<ol>

<li>uno</li>

<li>dos</li>

<li>tres</li>

<ul>

<li>tres.uno</li>

<li>tres.dos</li>

<ul>

<li>tres.dos.uno</li>

<li>tres.dos.dos</li>

</ul>

<li>tres.tres</li>

</ul>

<li>cuatro</li>

</ol>

* **Listas de definiciones <dl>.**

Cada entrada <dl> en la lista tiene dos partes:

* El término que se define, encabezado por la etiqueta <DT>
* La definición, encabezada por la etiqueta <DD>.

<dl>

<dt>termino1</dt>

<dd>definición1</dd>

<dt>termino2</dt>

<dd>definición2</dd>

</dl>

Salida:

termino1

definición1

termino2

definición2

1. **Enlaces <a>.**

* **Enlace con otro documento.**

Son referencias a otro documento o parte de el.

<a href=“url o marca de destino”>.......</a>

href: dirección url con la que se enlaza.

* **Marcadores de destino de un enlace.**
* *Define un punto de enlace en una página:*

<a name=nombre del enlace></a>

*name*: define un nombre en el documento que puede ser destino de un hiperenlace.

* *Hiperenlace hacia otra pagina:*

<a href=“http://www.microsoft.com”>La Web de Microsoft</a>

* *Establecer un marcador dentro de una pagina:*

<a name=“marca1”></a>

* *Hiperenlace hacia dicho marcador desde la misma pagina:*

<a href=“#marca1”>Documentación publica</a>

* *Hiperenlace a una dirección E-Mail:*

<a href=“mailto: dirección de E-mail”>Texto del enlace </a>

Ejemplo:

<html>

<head>

<title>ejemplo 3</title>

</head>

<body>

   <a name=“arriba”><h1>página de enlaces</h1></a>

   <a href=“#abajo”>ir abajo</a><br>

   <a href=“ej4.html”>ir a ejemplo 4</a><br><br>

   <a href=“http://www.webestilo.com/”>ir a webestilo</a><br>

   <br>.<br>.<br>.<br>.<br>.<br>.<br>.<br><br>

   <a name=“abajo”><br></a>

   <a href=“#arriba”>ir arriba</a>

</body>

</html>

1. **Agrupación de contenido.**

* **<div>.**

Permite agrupar varios bloques de texto en uno solo, heredando todos ellos la alineación especificada mediante el parámetro.

align=(left|center|right)

<html>

<body>

<div id=”Header” align=”center”>

<h2>Hola!</h2>

<p>Me llamo Aníbal</p>

</div>

</html>

* **<span>.**

Sirve para identificar un elemento en línea y de esta forma asignarle nombre, clase y estilo. Podemos cambiar el tamaño de la fuente, el color, el alto de la línea, la posición del texto, etc…

<html>

<body>

<div id=”Header” align=”center”>

<h2>Hola!</h2>

<p>Me llamo Aníbal</p>

</div>

<div id=”Content” align=”center”>

<h2>Contenido</h2>

<p>Bla bla bla bla

<span style=”font-size: 16pt”>

</p>

</div>

</html>

1. **Imágenes y mapas sensibles.**

* **<img>.**

<img> Incluir imagen. Coloca una imagen en la página. Uso:

<img align=alineación alt=texto\_alternativo src= url\_de\_la\_imagen border=ancho del borde

width=ancho en puntos o porcentaje(%) height=alto en puntos o porcentaje(%)>

align: Alineación de la imagen respecto a la base de la línea actual.

top : La parte superior de la imagen se alinea con el elemento mayor del texto

middle: La imagen se centra respecto a la línea.

bottom: La imagen se apoya sobre la base de la línea.

right: La imagen se coloca a la derecha de la ventana y es rodeada por el texto

left: La imagen se coloca a la izquierda de la ventana y es rodeada por el texto

Ejemplo:

<img src=“imagen1.jpg” align=“left” border=1>

Nota: una imagen puede ser utilizada dentro de un hiperenlace en lugar del texto normal.

<a href=”http://www.ole.es”><img src=“imagen1.jpg” border=”1”></a>

* **<map> & <área>.**

La estructura de mapas sensibles nos permite asignar diferentes áreas de una imagen a diferentes hipervínculos. Consta de dos elementos :

* *Una estructura de mapa:*

<map name= >

<area shape= coords= href= )>

.......

</map>

* <MAP NAME= > Etiqueta de apertura del Mapa, define el nombre de este.
* <AREA SHAPE= COORDS= (HREF= | NOHREF)> Definición de la zona activa del mapa.
* SHAPE Define la forma de la zona activa.
* COORDS Coordenadas de la zona activa.

shape=rect coords=“left-x, top-y, right-x, bottom-y”

shape=circle coords=“center-x, center-y, radius”

shape=poly coords=“x1,y1, x2,y2, x3,y3, ...”

* HREF Hiperlink de la zona activa.
* *Un atributo para la etiqueta <img> en el que referiremos al mapa:*

<img src = usemap =“#nombre del mapa”>

Ejemplo:

<map name="mapa1">

<area shape="circle" coords="58, 84, 19" href="http://www.google.com">

<area shape="rect" coords="28,159,62,198" href="http://www.wikipedia.com">

</map>

<img src="/images/hidra.gif" usemap="#mapa1">

1. **Tablas.**

Agrupación de casillas en filas y columnas, que nos permiten un diseño de página mas preciso.

Una tabla se compone de los siguientes elementos:

* **Definición de la tabla <table>**.

<table></table>. Señala el inicio y final de una tabla. Sus atributos son:

* **Align,** establece la alineación de la tabla o texto mediante ALIGN=LEFT o ALIGN=RIGHT.
* **Bgcolor**, establece el color de fondo de las celdas de la tabla.
* **Border**, determina el ancho del borde en pixeles.
* **BorderColor**, asigna un color al borde.
* **BorderDark**, determina el color de la parte oscura de un borde de 3 dimensiones.
* **BorderLight**, asigna el color de la parte clara de un borde de 3 dimensiones.
* **Caption**, especifica el titulo para la tabla.
* **Cellpadding**, establece la cantidad de espacio libre junto al contenido de una celda.
* **Cellspacing**, asigna la cantidad de espacio entre las celdas de una tabla.
* **Width**, determina el ancho de la tabla en píxeles o en un porcentaje.
* **Definición de cabecera <th>.**

<th></th>. Indica al navegador cliente que exhiba el texto como un encabezado en la primera fila de una tabla. Atributos principales:

* **Colspan,** especifica el número de celdas que cubre el encabezado.
* **Align**, determina la posición del texto del titulo.
* **Definición de fila <tr>.**

<tr> </tr>. Indica al navegador que exhiba el texto dentro de una fila; puede también interpretarse como la etiqueta que define filas.

* **Align,** alineación del texto/objetos en toda la fila.
* **Definición de casilla <td>.**

La etiqueta de datos de la tabla, es la que identifica a las columnas o celdas específicas de una tabla. Atributos principales:

**Align**, alineación del texto/objeto de la celda

**Bgcolor**, color de fondo de la celda

**Background**, imagen de fondo de una celda

**Width**, ancho de la celda/columna con respecto al ancho de la tabla

Solo precisa definir el ancho en la primera celda de la columna.

Dentro de una celda, se puede insertar desde texto o un gráfico hasta una tabla entera.

Ejemplo:

<table width=80% align=center border=3>

<tr>

<th colspan=3>Directorio Telef&oacute;nico</th>

</tr>

<tr align=center>

<td width=40% bgcolor="#C0C0C0"> Nombre </td>

<td width=30% bgcolor="#C0C0C0"> Teléfono </td>

<td width=30% bgcolor="#C0C0C0"> E-Mail </td>

</tr>

<tr>

<td>Jos&eacute; Rodr&iacute;guez </td>

<td> 223454 </td>

<td> jose@prueba.com.bo</td>

</tr>

<tr>

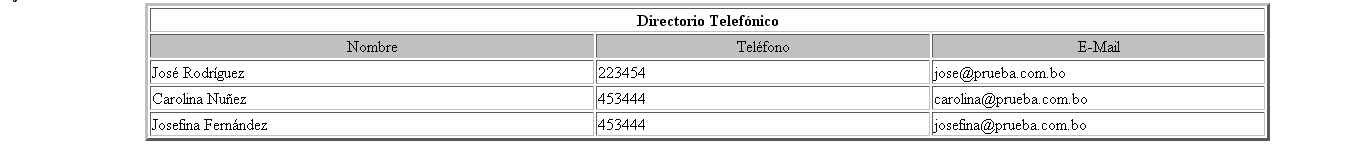
<td> Carolina Nu&ntilde;ez </td>

<td> 453444 </td>

<td>carolina@prueba.com.bo</td>

</tr>

</table>



* **<thead>, <tfoot>, <tbody>.**

Son etiquetas que agrupan el contenido de una tabla, para, por ejemplo, darle con CSS el mismo formato: El cuerpo de tabla <tbody> , puede haber tantos como se quiera.

Tanto a la cabecera de la tabla <thead> como el pie de tabla <tfoot> son opcionales y sólo puede haber uno de cada por tabla. En el código fuente, la etiqueta <tfoot> se encuentra situada antes del primer <tbody>, aunque los navegadores la muestran al final de la tabla.

* **Columnas <col> y grupos de columnas <colgroup>.**

Aunque las celdas de una tabla <td> y <th> estén organizadas en filas <tr> y grupos de filas <tbody>, <thead> y <tfoot>, también existen dos etiquetas que permiten hacer referencia a las columnas de una tabla: las etiquetas <col /> columna y <colgroup> grupo de columnas.

La etiqueta <col /> permite hacer referencia a una columna y la etiqueta <colgroup> permite definir grupos de columnas (de manera similar a como la etiqueta <tbody> define grupos de filas).

Las etiquetas <col /> y <colgroup> se encuentran situadas al principio de la tabla, después de la etiqueta <caption>.

El ejemplo siguiente muestra la situación de las etiquetas <col /> y <colgroup> en una tabla.

Ejemplo

<table width="80%" align="center" border="3">

<colgroup bgcolor="red" align="center">

<col />

<col />

</colgroup>

<thead>

<tr>

<th colspan="3">Directorio Telef&oacute;nico</th>

</tr>

</thead>

<tfoot>

<tr>

<th colspan="3">TFoot</th>

</tfoot>

<tbody>

<tr align="center">

<td width="40%" bgcolor="#C0C0C0"> Nombre </td>

<td width="30%" bgcolor="#C0C0C0"> Teléfono </td>

<td width="30%" bgcolor="#C0C0C0"> E-Mail </td>

</tr>

<tr>

<td>José; Rodríguez </td>

<td> 223454 </td>

<td> jose@prueba.com.bo</td>

</tr>

<tr>

<td> Carolina Núñez </td>

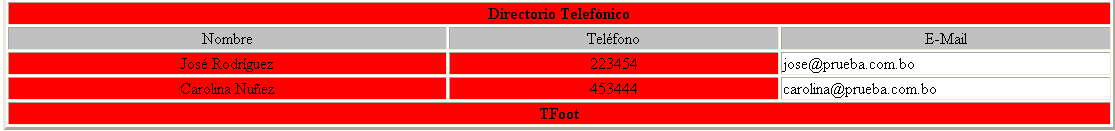
<td> 453444 </td>

<td>carolina@prueba.com.bo</td>

</tr>

</tbody>

</table>



1. **Formularios.**

Los formularios nos permiten dentro de una página Web solicitar información al visitante para después procesarla.

En el formulario podremos solicitar diferentes datos (campos), cada uno de los cuales quedará asociado a una variable. Una vez se hayan introducido los valores en los campos, el contenido de estos será enviado a la dirección (URL) donde tengamos el programa que pueda procesar las variables.

* **<form>.**

La declaración del formulario se pone entre las directivas <form></form>. En el interior de la declaración se indican los elementos (variables) de entrada. La directiva <form> tiene los parámetros action , method y [enctype].

Uso:

<form action=url

method=método de envío

enctype=método de codificación de la información>

campos de entrada, selección y áreas de texto

…………

………..

</form>

* *action = “programa”*

Indica el programa que va a “tratar” a las variables que se envíen con el formulario.

* *method = POST / GET. "*

Indica el método según el que se transferirán las variables.

El valor get es el valor por defecto. Si no concretamos el method, la información se enviará a través de este medio.

“get” indica que los datos enviados se adjuntarán en la barra de direcciones del cliente, al final de la url correspondiente y después de un signo de interrogación de cierre. Si se envía más de un dato, éstos irán separados por el símbolo &.

Un ejemplo de un formulario enviado por el method=“get” sería el siguiente: http://www............?nombre1=valor1&nombre2=valor2

El valor post indica que el método de envío no se hará a través de la url, sino formando parte del cuerpo de la petición.

* *Enctype =*

Indica el modo de cifrado de la información. La forma puede ser de varios tipos, aunque el más común es que la información se envíe como texto plano (enctype="text/plain").

Ejemplo

<form action = "mailto: prufo@olemail.es" method = post enctype="text/plain">

* **<input>.**

Para la introducción de las variables se utiliza la directiva **<input>**. Esta directiva tiene el parámetro **type** que indica el tipo de variable a introducir y **name** que indica el nombre que se le dará al campo. Cada tipo de variable tiene sus propios parámetros.

* *type= text name = campo*

Indica que el campo a introducir será un texto. Sus parámetros son:

* maxlenght = numero

Numero máximo de caracteres a introducir en el campo.

* size = numero

Tamaño en caracteres que se mostrará en pantalla.

* value = “texto”

Valor inicial del campo. Normalmente será “ “, o sea, vacío.

Ejemplo:

tu usuario:<input type = text name = “nombre” size = “30” >

Salida



* *type = password name = campo*

Indica que el campo será una palabra de paso. Mostrará asteriscos (\*) en lugar de las letras escritas. Sus parámetros opcionales son los mismos que para text.

Ejemplo:

tu contraseña: <input type = password name = clave size = 8 >

Salida



* *type = checkbox name = campo*

El campo se elegirá marcando una casilla. Se permite marcar varias casillas. Los valores de las casillas serán indicados por:

* value = “valor”
* checked :La casilla aparecerá marcada por defecto.

Ejemplo:

Información a recibir:<br>

<input type = checkbox name = archivo value = "html" > manual de html

<input type = checkbox name = archivo value ="editor" > programa editor

<input type = checkbox name = archivo value ="ejemplo" > archivo de ejemplos

Salida:



* *type = radio name = campo*

El campo se elegirá marcando una casilla. Solo permite marcar una sola de las casillas. Los valores de las casillas serán indicados por: value = “valor”

Ejemplo:

Tu edad : <input type = radio name = edad value = "-20" > menos de 20 años

<input type = radio name = edad value = "20-40" > entre 20 y 40 años

<input type = radio name = edad value = "+40" > mas de 40 años

Salida:



* *type = hidden name = campo*

El usuario no puede modificar su valor, ya que el campo no es visible. Se manda siempre con el valor indicado por el parámetro: value = “valor” .

Ejemplo

<input type = hidden name = lugar value = "pagina personal" >

* *type = submit*

Representa un botón. Al pulsar este botón la información de todos los campos se envía al programa indicado en <FORM>. Tiene el parámetro value = “texto” que indica el texto que aparecerá en el botón.

Ejemplo:

<input type = submit value = "enviar" >

Salida:



* *type = reset*

Representa un botón. Al pulsar este botón se borra el contenido de todos los campos. El parámetro value = “texto” indica el texto que aparecerá en el botón.

Ejemplo:

<input type = reset value = "borrar" >

Salida:



* **<button>.**

Esta etiqueta proporciona un método único para la implementación de cualquier tipo de botón de formulario. Sus principales atributos son:

type= " tipo ", que puede tomar los ya conocidos valores submit (por defecto), reset y button.

name= " nombre ", que asigna un nombre identificador único al botón.

value= " texto ", que define el texto que va a aparecer en el botón.

La principal ventaja que aporta estas etiquetas es que ahora vamos a poder introducir dentro de ellas cualquier elemento de HTML, como imágenes y tablas.

Ejemplos:

<form action="cgi-bin/control.exe" method="post" enctype="text/plain" name="miform">

<button name="boton\_1" type="button">

<table width="10" cellspacing="0" cellpadding="2" border="1">

<tr>

<td>uno</td>

<td>dos</td>

</tr>

<tr>

<td>tres</td>

<td>cuatro</td>

</tr>

</table>

</button>

</form>

* **<select>.**

Sirve para insertar un menú desplegable con varias opciones. <select> </select>. Parámetros:

* name = campo

Nombre del campo

* multiple

Permite seleccionar más de un valor para el campo.

* disabled

La lista solo se visualiza. El usuario no puede seleccionar.

* **<option>.**

Las diferentes opciones de la lista se indican con la directiva <OPTION>. Sus parámetros son:

* value = valor

Valor a enviar si se selecciona el elemento. Si no esta se devuelve el texto de la opción.

* selected

La opción aparece seleccionada.

* disabled

La opción no se puede seleccionar

Ejemplo:

Como encontraste mi página:

<select name = donde >

<option>de casualidad

<option>por el buscador ole

<option>por el buscador yahoo

<option>me la comentaron

</select>

Salida:



* **<textarea>.**

Representa un campo de texto de múltiples líneas. Normalmente se utiliza para que se incluyan en él comentarios.

La directiva usada es <textarea></textarea>. Sus parámetros son:

name = campo

Nombre del campo.

cols = num.

Numero de columnas de texto visibles.

rows = num.

Numero de filas de texto visibles.

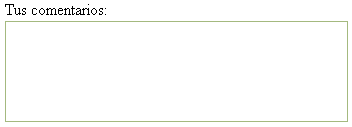
Ejemplo:

Tus comentarios:

<br>

<textarea name = comentario rows = 5 cols = 40></textarea>

Salida:



* **<label>.**

El elemento <label> se usa para especificar una etiqueta para otro elemento en el documento. El elemento requiere un comando de apertura y uno de cierre. El elemento es un elemento inline.

Ejemplo:

<form action="cgi-bin/micontrol.exe" method="post" enctype="text/plain" name="un ejemplo más">

<label>

<input type="checkbox" name="email">

Le deseamos un feliz año nuevo

</label>

</form>

* **<fieldset>.**

Sirve para agrupar los controles y sus títulos por temas.

* **<legend>.**

Sirve para asignar un rótulo a un fieldset.

<form>

<fieldset>

<legend>Personalia:</legend>

Name: <input type="text"><br>

Email: <input type="text"><br>

Date of birth: <input type="text">

</fieldset>

</form>

1. **HTML5.**

Esta nueva versión aporta nuevos elementos enfocados a distintas tareas como la presentación, el diseño, inclusión de audio y video, mejorar formularios y almacenar datos de sesión.

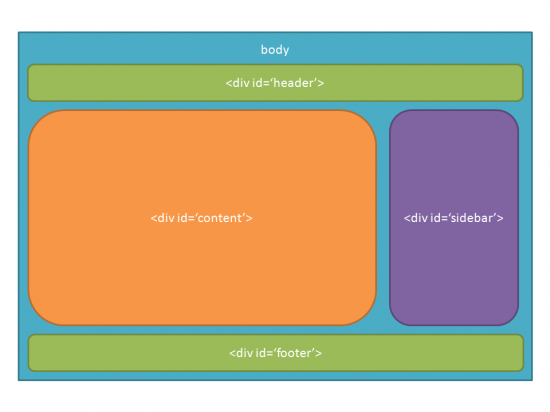
Su objetivo es convertirse en una plataforma de desarrollo que integre todas las tecnologías web, combinando HTML, CSS y JS, y diferentes formatos de audio, video y animaciones.

HTML5 pretende ser una solución a esto incorporando una Application Program Interface (API), es decir, una biblioteca de funciones amplia para que todos los navegadores se comporten igual.

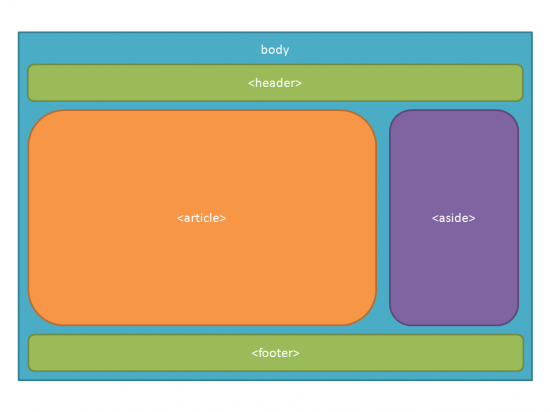
Características:

* Se permiten mayúsculas en las etiquetas.
* La etiqueta de cierre es opcional en los elementos vacíos.
* Es opcional establecer un valor a los atributos.
* Las comillas son opcionales en los atributos.

**Estructura básica en HTML 4:**



**Estructura de HTML 5:**



**<header>** para indicar el bloque contenedor de la cabecera (bloque superior de información) que suele haber siempre en las páginas, también se puede usar en otros puntos del sitio.

**<nav>** representa la sección de la página orientada a la navegación. Este elemento es el que agrupa los enlaces. No todos los enlaces han de estar agrupados en este elemento, sino sólo aquellos grandes bloques que incluyan enlaces.

**<section>** definición de una sección del sitio. Es la sección general dentro de un documento y suele contener los elementos <h1-h6>.

**<article>** nueva etiqueta para definir bloques de contenidos.

**<aside>** es la barra lateral. Digamos que, por norma general, este bloque incluirá la zona de navegación del sitio web, las categorías, opciones o como queramos llamarlo.

**<footer>** Por norma general los sitios web tienen un pie de página en el que se indican datos legales y similares, y esta podría ser la principal función, aunque no está pensado para que sea la única..

**<hgroup>** agrupar un conjunto de encabezados en un sólo bloque para que puedan ser manejados como una unidad. Cuando mencionamos la palabra encabezado nos estamos refiriendo nada más y nada menos que las etiquetas que van de h1 a h6, y que nos sirven para darle un formato a los títulos de nuestros artículos, entradas o nombres lo que hace resaltarlos del resto del texto.

Ejemplo:

<!DOCTYPE html>

<html>

<head>

<title> Una página HTML simple </title>

</head>

<body>

<header>Header

<nav>

<a href='#'>Algún enlace de navegación</a>

<a href='#'>Algún enlace de navegación adicional</a>

<a href='#'>Un tercer enlace de navegación</a>

</nav>

</header>

<section>

<article>

<p> Esta es una sección importante del contenido de la página. Tal vez una publicación en blog.</p>

<aside>

<p> Este es un aparte de la primera publicación en blog. </p>

</aside>

</article>

<article>

<p>Esta es una sección importante del contenido de la página. Tal vez una publicación en blog.</p>

</article>

</section>

<footer>Footer</footer>

</body>

</html>

* 1. **Contenido editable.**

Esta característica permite editar cualquier parte del texto contenido dentro del elemento con este atributo. Solo debes añadir:

contenteditable="true"

Al elemento que quieras hacer editable, por ejemplo:

<body>

<h2> Lista </h2>

<ul contenteditable="true">

<li> Elemento #1</li>

<li> Elemento #2

<li> Elemento #3</li>

</ul>

</body>

* 1. **Atributo requerido.**

Los formularios permiten el atributo ‘required‘ para especificar que un campo es requerido.  Los datos no serán enviados si el campo marcado como requerido está vacío.

Puedes utilizar este atributo de dos formas:

<input type="text" name="Input1" required>

<input type="text" name="Input1" required="required">

* 1. **Campo email.**

Si se aplica el tipo email (type=”email”)en un campo de un formulario, el navegador solo permitirá ingresar direcciones de email en dicho campo.

Un ejemplo sería:

<body>

<form action="" method="get">

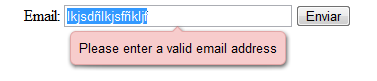
<label for="email">Email:</label>

<input id="email" name="email" type="email" />

<button type="submit"> Enviar </button>

</form>

</body>



* 1. **Atributo placeholder.**

Permite rellenar el campo con un valor predeterminado y una vez que el usuario hace clic en el campo este valor desaparece.

<input name="email" type="email" placeholder="dacj@trazos-web.com" />



* 1. **Expresiones regulares.**

Si deseas que  en un campo de un formulario solo se puedan escribir solo un número máximo de caracteres, o solo mayúsculas, etc., esto lo puedes hacer mediante expresiones regulares utilizadas directamente en el código HTML:

<form action="" method="post">

<label for="username">Nombre de Usuario: </label>

<input type="text"

name="username"

id="username"

placeholder="4 <> 10"

pattern="[A-Za-z]{4,10}"

required>

<button type="submit">Enviar </button>

</form>

* 1. **<audio>.**

<audio controls>

<source src="sound.ogg" type="audio/ogg" >

<source src="sound.mp3" type="audio/mp3" >

</audio>

* 1. **<video>.**

<video width="640" height="360" src="/demo/google\_main.mp4?2" autobuffer>

<div class="video-fallback">

<br>You must have an HTML5 capable browser.

</div>

</video>

* 1. **<embed>.**

Sirve para contenido incrustado pero no nativo, sino ejecutado por plugins como el de Flash.

<embed src="mid/buenofeomalo.mid" autostart="true" loop="true" hidden="true"></embed>

<embed src="http://www.youtube.com/v/SzSwnbxb9TY?fs=1&hl=es\_ES&rel=0&color1=0x234900&color2=0x4e9e00" type="application/x-shockwave-flash" allowscriptaccess="always" allowfullscreen="true" width="640" height="385"></embed>

* 1. **<canvas>.**

Espacio para gráficos generados en el momento de la visualización de la web. Es una forma de dibujar de manera programada mediante Javascript

* 1. **<command>.**

Para mostrar comandos que el usuario puede ejecutar:

<menu>

<command onclick="alert('Hola Mundo')">¡Click Aqui!</command>

</menu>

* 1. **<datagrid>.**

Ideal para mostrar un árbol de datos o una tabla tabulada. El usuario, a diferencia de las tables normales, puede seleccionar filas, columnas, celdas, editar celdas, borrar filas, etc. <datalist> junto con el nuevo atributo list para los <input /> puede ser usado para crear comboboxes.

<datagrid>

<table>

<tr><td>Jones</td><td>Allison</td><td>A-</td><td>B+</td><td>A</td></tr>

<tr><td>Smith</td><td>Johnny</td><td>A</td><td>C+</td><td>A</td></tr>

<tr><td>Willis</td><td>Sydney</td><td>C-</td><td>D</td><td>F</td></tr>

<tr><td>Wilson</td><td>Frank</td><td>B-</td><td>B+</td><td>A</td></tr>

</table>

</datagrid>

* 1. **<details>.**

Muestra información adicional si el usuario lo demanda:

<details>

<summary>Copyright 1999-2011.</summary>

<p>All pages and graphics on this web site are the property of the company Refsnes Data.</p>

</details>

* 1. **<dialog>.**

Tag para definir diálogos y conversaciones entre varias personas, por ejemplo transcripciones del Chat.

<dialog>

<dt> Bounty hunter

<dd> You're wanted, Wales.

<dt> Outlaw Josey Wales

<dd> Reckon I'm right popular. You a bounty hunter?

<dt> Bounty hunter

<dd> A man's got to do something for a living these days.

<dt> Outlaw Josey Wales

<dd> Dyin' ain't much of a living, boy.

</dialog>

* 1. **<figure>.**

Podrás usarlo para asociar con un caption un contenido incrustado, como por ejemplo gráficos o vídeo.

<p><a href="#1">Figure 1</a> provides the JavaScript code for creating an alert box:</p>

<figure id="1">

<figcaption>Figure 1. JavaScript Alert Box.</figcaption>

<pre><code>alert('Hello!');</code></pre>

</figure>

* 1. **<keygen>.**

Define una clave generada en un formulario. Cuando el formulario es enviado, la clave privada es almacenada y la clave publica se envía al servidor.

<form action="mailto:direccion@correo.com" method="post">

Username: <input type="text" name="usr\_name" />

Encryption: <keygen name="security" />

<input type="submit" />

</form>

* 1. **<meter>.**

Usado para representar medidas, por ejemplo el tamaño del disco usado.

<meter value="2" min="0" max="10">2 out of 10</meter><br />

<meter value="0.6">60%</meter>

* 1. **<mark>.**

Esta etiqueta sirve para resaltar un texto. Puede ser útil cuando un usuario realiza una búsqueda, para resaltar la palabra o frase buscada en el sitio web:

<h3> Resultados de la búsqueda </h3>

<p>Seguro ya habrás escuchando en los últimos meses del

<mark>HTML5</mark>, el nuevo estándar.

</p>

* 1. **<output>.**

Define salidas en un formulario:

<html>

  <body>

    <form action="form.asp" method="get" name="avgform">

      <output name="avg"></output>

    </form>

  </body>

</html>

* 1. **<progess>.**

Implementación de una barra de progreso durante la realización de un proceso o tarea:

Downloading now. Progress...

<progress value="250" max="1000">

<span id="downloadProgress">35</span>%

</progress>

* 1. **<source>.**

Define recursos multimedia:

<audio controls="controls">

  <source src="song.ogg" type="audio/ogg" />

  <source src="song.mp3" type="audio/mpeg" />

Your browser does not support the audio element.

</audio>

* 1. **<summary>.**

Define una cabecera o un detalle de algún elemento:

    <details>

      <summary>HTML 5</summary>

Through this document you will get great assistance to use summary tag and will teach you effective ways to learn about HTML 5 easily.

    </details>

* 1. **<time>.**

Información sobre tiempos o fechas:

    We arrived at <time>09:00</time>

**T3. CSS. Hojas de estilo.**

1. **Introducción.**

Cascading Style Sheets es un lenguaje de presentación (no de marcas). Está ligado a los lenguajes de marcas ya que a día de hoy resulta imprescindible para crear sitios web con calidad.

El **objetivo principal** de CSS es manejar el aspecto y formato de los documentos liberando de esta forma a HTML de las tareas de presentación.

Esta separación entre contenidos y presentación supone muchas ventajas y evita la repetición innecesaria de código. Facilita la generación de código y su mantenimiento y mejora la legibilidad de los documentos.

1. **Estructura.**

Podemos diferenciar tres partes en la sintaxis de las reglas de estilo: el selector, la propiedad y el valor.

selector {propiedad:valor;}

* El selector identifica los tags a los que debe de aplicarse la regla de estilo.
* La propiedad es el nombre de la característica a modificar del tag .
* El valor es la medida de esa característica.
* Se pueden modificar varias propiedades.

selector {propiedad1:valor1;propiedad2:valor2;..}

1. **Inclusión en HTML.**

* Definir CSS en un archivo externo.

En este caso, todos los estilos CSS se incluyen en uno, o varios, archivos de texto plano, cuya extensión es .css, que las páginas HTML enlazan mediante el elemento <link> de la cabecera del fichero HTML.

Puede hacerse de dos modos diferentes:

* *Mediante enlaces.*

<link rel="stylesheet" href="estilos.css" type="text/css" />

* *Importando el fichero CSS.*

<style type="text/css" media="screen">@import "estilos.css";</style>

Y el archivo estilos.css seria un conjunto de descriptores

*body {background-color: white; color: black; font-family: arial;}*

* Incluir CSS en el documento HTML (incrustado).

Este método se emplea cuando se definen pocos estilos o cuando se quieren incluir estilos específicos en una determinada página HTML que completen los estilos globales de todas las páginas del sitio web.

Tiene el inconveniente de que para modificar los estilos definidos, es necesario modificar todas las páginas que incluyen el estilo que se va a cambiar.

<head>

*<style type="text/css">*

*body {background-color: white; color: black; font-family: arial;}*

*</style>*

</head>

* Incluir CSS en los elementos HTML.

El último método para incluir estilos CSS en documentos HTML es el peor y el menos utilizado, ya que para modificar un formato hay que cambiar todos los elementos que estén asociados a él.

Solamente se utiliza en determinadas situaciones en las que se debe incluir un estilo muy específico para un solo elemento concreto.

*<h1 style="color: red; font-size: 30px; text-decoration: underline">CSS en línea</h1>*

1. **Anidamiento de selectores.**

* Comenzamos con el selector mas simple, que afectaría a todas las etiquetas h1:

*h1 {color: red}*

* Si nos interesa que afecte a los tag <h1> que se encuentran dentro de otros tipos de tag, como por ejemplo, <div>. En estos casos podríamos poner:

*div h1 {color: red}*

*<head>*

*<style type="text/css">*

*h3{color: blue}*

*div h3 span{color: green; font-size: 125%}*

*Div h3 {color: yellow}*

*</style>*

*</head>*

*<body>*

*<h1>Titulares de Prensa</h1>*

*<div>*

*<h3>Agricultura afirma:*

*<span>El precio de la patata se dispara</span>*

*</h3>*

*<p>El Ministro de Agricultura afirmó el pasado jueves que...</p>*

*</div>*

*<h3>prueba</h3>*

*</body>*

* Podríamos querer que la regla sólo afectase a los tag <h1> que se encuentran alojados en un contenedor <div> concreto, no en todos los contenedores <div>.

*#contenedor h1{font-size: 0,9em;*

*font-weight: bold;*

*color: #0000FF;*

*margin-bottom: 10px;}*

*<div id="contenedor">*

*<h1>Este es el titular de la noticia</h1>*

*<p>Este es el contenido de la noticia</p>*

*</div>*

La forma de crear selectores para un tag a través de su identificador es:

*#identificador*

* **Class e Id:**
* *id = name.* Este atributo asigna un nombre a un elemento. Este nombre debe ser único en un documento.
* *class = lista de cdata.* Este atributo asigna un nombre de clase o un conjunto de nombres de clase a un elemento. Se puede asignar el mismo nombre o nombres de clase a cualquier número de elementos. Los nombres de clase múltiples deben estar separados por caracteres de espacio en blanco.
  + Crear una clase que podamos aplicar a los tag mediante el atributo class:

*.primerparrafo {font-size: 75%;}*

*<p class=”primerparrafo”>Este es el primer árrafo</p>*

* Crear un estilo asociado a una clase:

*p.azuloscuro {color: darkblue;}*

*<p class="azuloscuro">Este es el primer parrafo</p>*

* A un mismo tag se le pueden aplicar dos o más clases dentro de su atributo class.

*<head>*

*<style type="text/css">*

*body {color: red;}*

*.azuloscuro {color: darkblue; font-size:100%}*

*.grande {font-size: 200%;}*

*</style>*

*</head>*

*<body>*

*<p>Este es el primer parrafo</p>*

*<p class="azuloscuro">Este es el segundo parrafo, el parrafo que viene despues. Como puedes ver, esta; escrito en color azul oscuro</p>*

*<p class="azuloscuro grande">Este es el tercer parrafo. Tambien estae; en color azul oscuro y además tiene la letra más grande</p>*

*</body>*

* Si quisiéramos que el orden de escritura de las clases no sea el cualificador de orden se puede usar *!important*

*.grande {font-size: 200% !important;} /\* Asi no nos preocupamos del orden\*/*

*.azuloscuro {color: darkblue; font-size:100%}*

Para que tomen precedencia sobre otras reglas de estilos que se puedan encontrar en una página web.

Un mismo tag puede estar afectado por varias reglas de estilo.

Algunas propiedades tienen la característica de herencia, pasan de los elementos padre a los elementos hijo.

*!important*

*td{*

*font-family: verdana, arial !important;*

*}*

*td.micelda{*

*font-family: monospace;*

*}*

*<table>*

*<tr>*

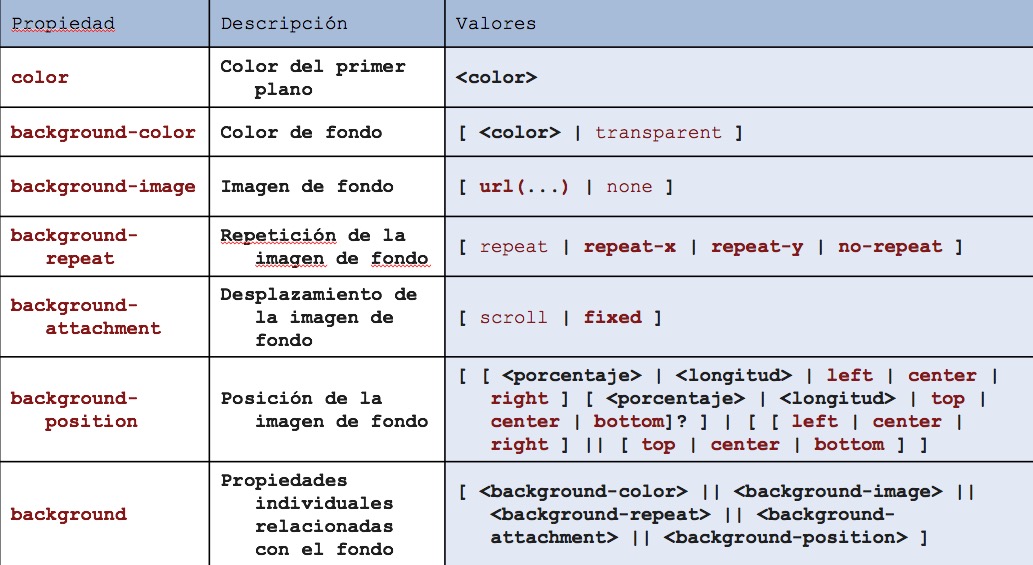
*<td class="micelda">Hola</td>*

*<td>23232</td>*

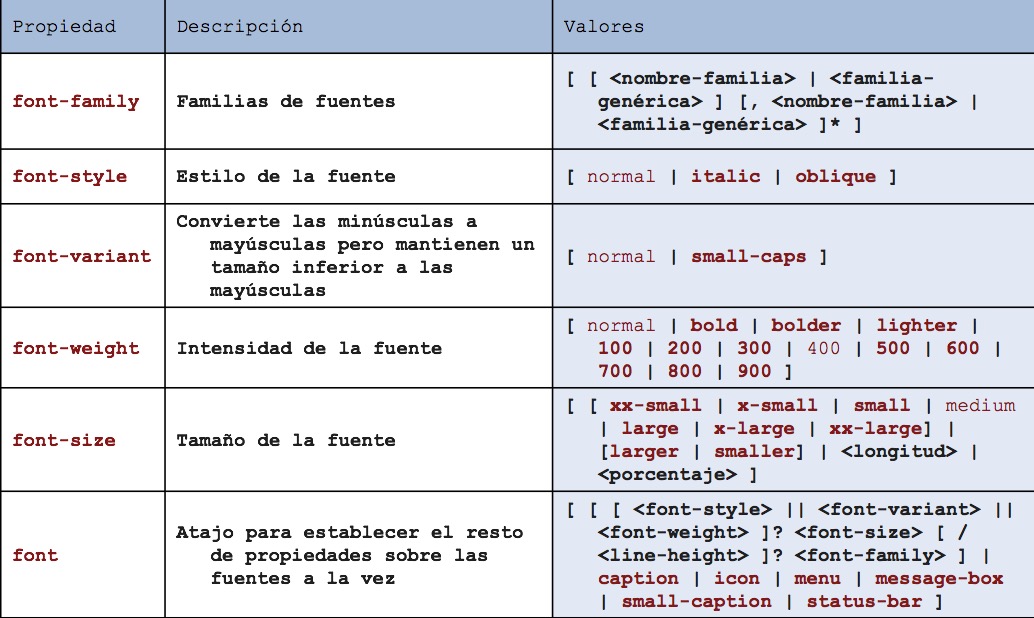
*</tr>*

*</table>*

1. **Propiedades CSS.**
   1. **Colores y fondo.**



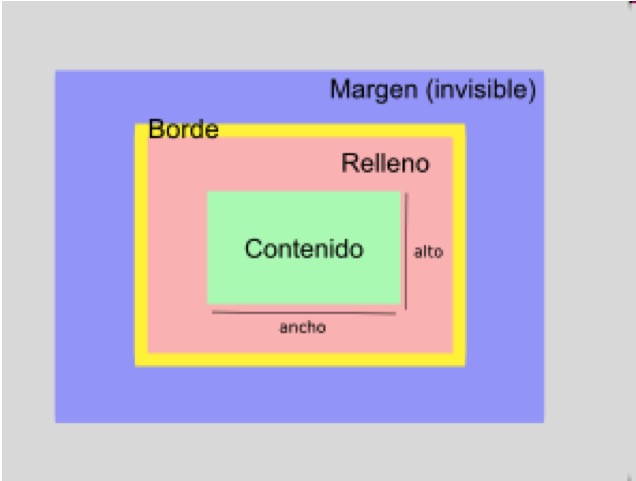
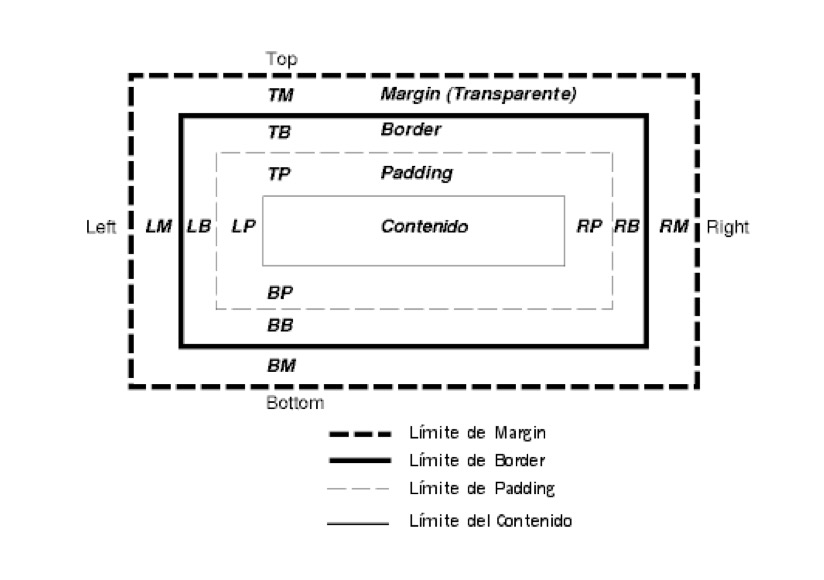
* 1. **Fuentes.**

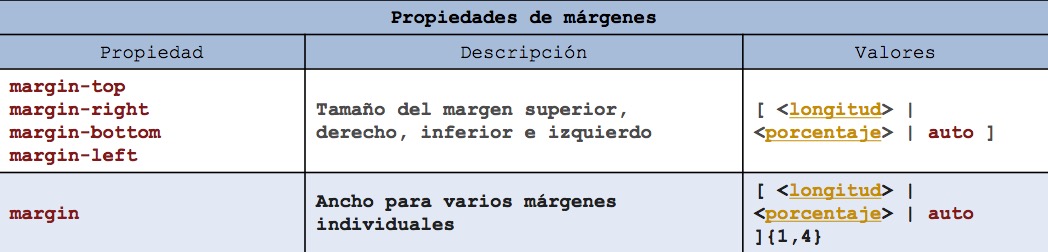


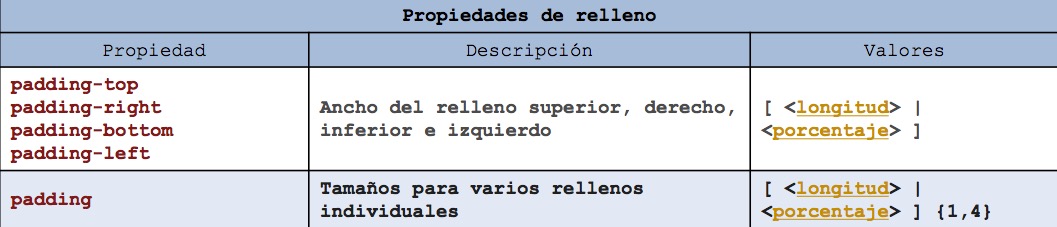
* 1. **Texto.**



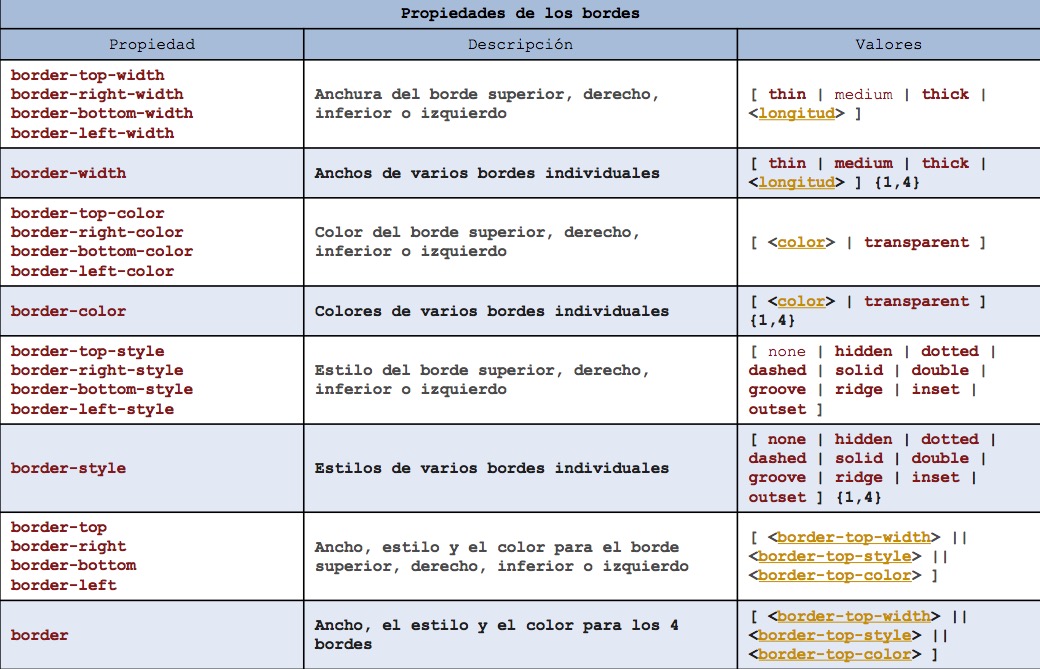
* 1. **Márgenes y rellenos.**

****

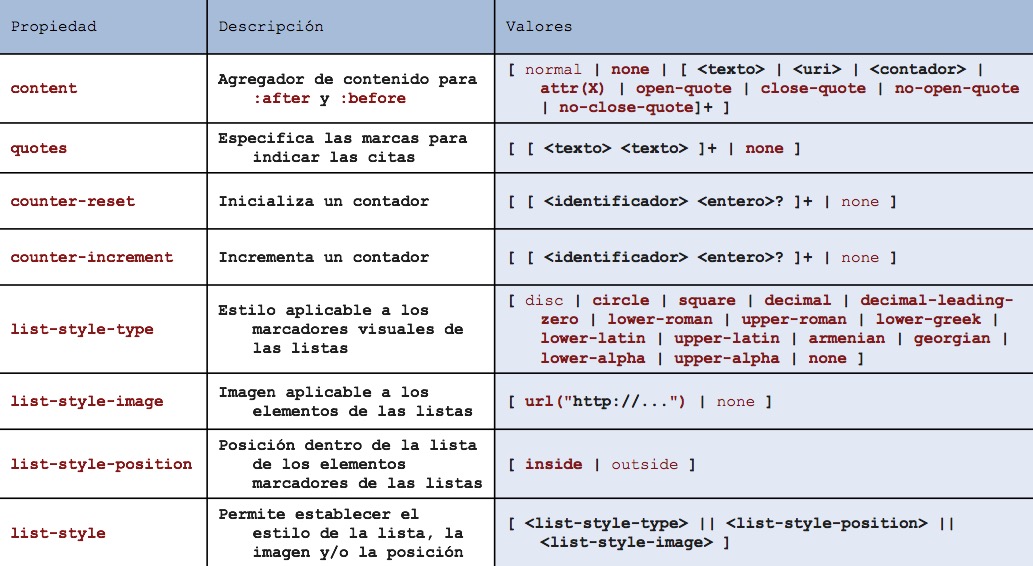
****

****

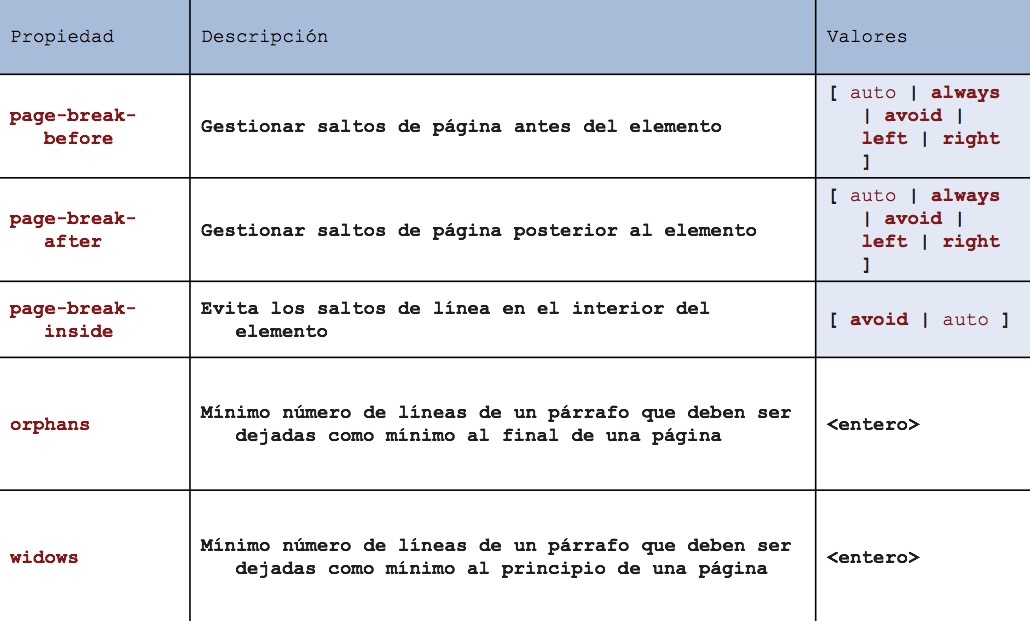
* 1. **Bordes.**



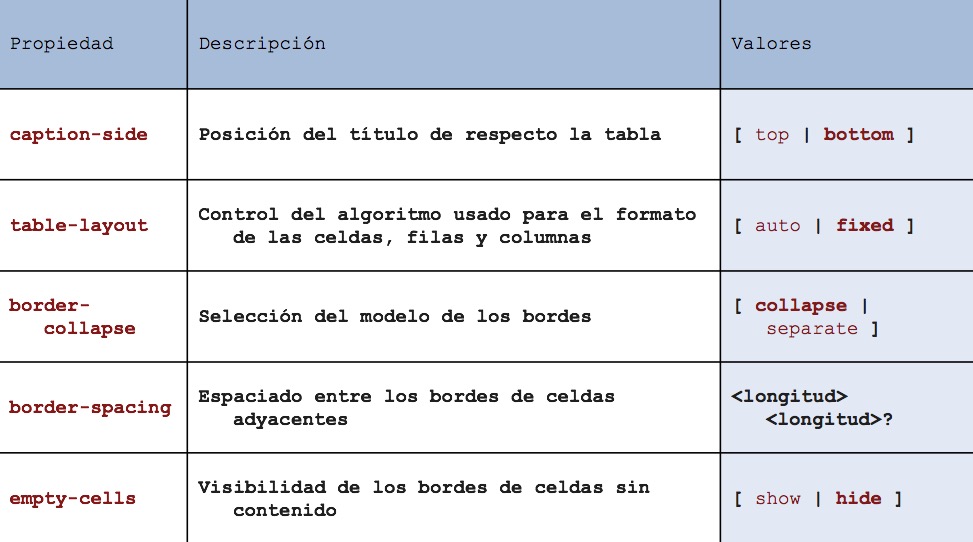
* 1. **Numeración y listas.**



* 1. **Medios paginados.**



* 1. **Tablas.**

****

* 1. **Formato visual.**

****

* 1. **Position.**
* *static.*

La caja es una caja normal, situada de acuerdo al flujo normal. Las propiedades 'top', 'right', 'bottom' y 'left' no se aplican.

* *relative.*

La posición de la caja se calcula de acuerdo al flujo normal (ésta se llama la posición en el flujo normal). Luego la caja se desplaza de modo relativo a su posición normal.

* *absolute.*

La posición de la caja (y posiblemente el tamaño) se especifica con las propiedades 'left', 'right', 'top', y 'bottom'. Estas propiedades especifican los desplazamientos con respecto al bloque de contención de la caja. Las cajas posicionadas absolutamente son quitadas del flujo normal. Esto significa que ellas no tienen ningún impacto sobre la composición de los hermanos siguientes. También, aunque las cajas posicionadas absolutamente tiene márgenes, no se cierran con ningún otro margen.

* *fixed.*

La posición de la caja se calcula de acuerdo al modelo 'absolute', pero además, la caja se fija con respecto a alguna referencia visual, la pantalla. La caja no es afectada por el scroll y debe definirse un Doctype.

* 1. **Float.**

Esta propiedad especifica si una caja debe flotar a la izquierda, derecha o no debe flotar en absoluto. Puede especificarse para los elementos que generan cajas que no están posicionadas absolutamente. Los valores de esta propiedad tiene los siguientes significados:

* *left*.

El elemento genera una caja de bloque que flota a la izquierda. El contenido fluye sobre el costado derecho de la caja, comenzando en la parte superior (sujeto a la propiedad 'clear'). La propiedad 'display' es ignorada, a menos que tenga el valor 'none'.

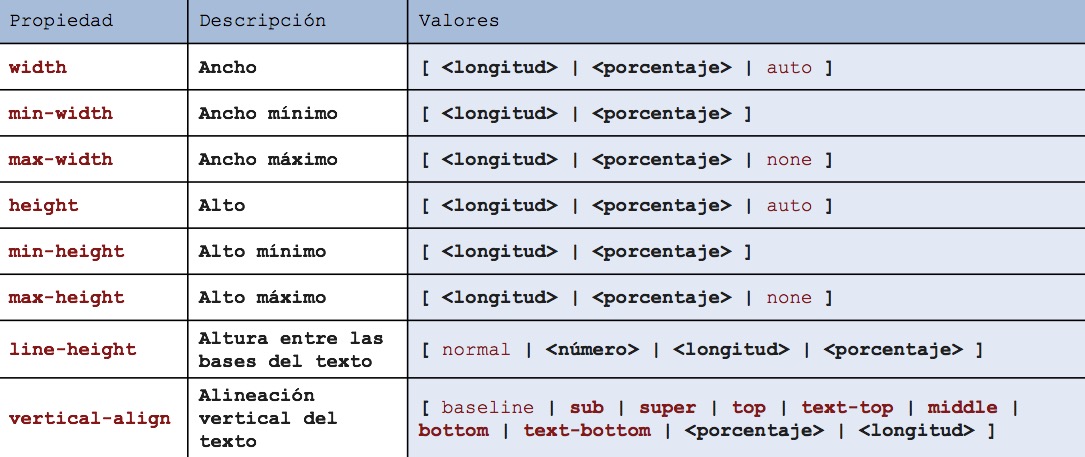
* *right.*

Igual que 'left', pero el contenido fluye sobre el costado izquierdo de la caja, comenzando en la parte superior.

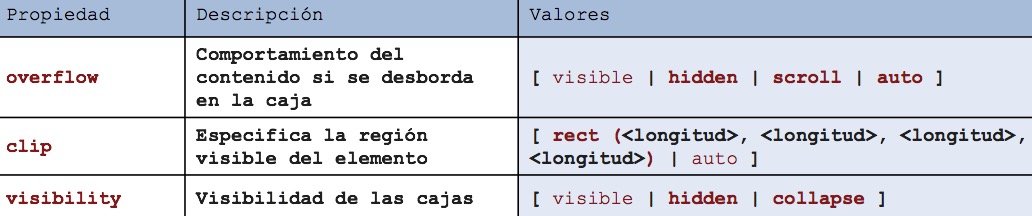
* *none*.

La caja no es flotante.

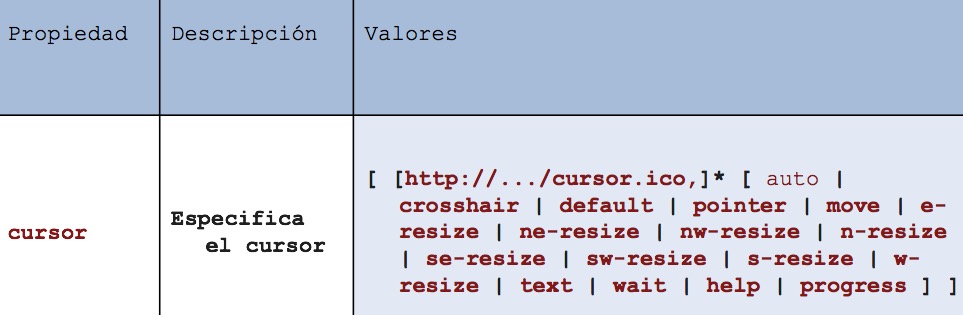
* 1. **Detalle visual.**

****

* 1. **Efecto visual.**

****

* 1. **Interface de usuario.**

****

* 1. **Medidas.**
* *Relativas:*

**Element (em**): Expresa el tamaño relativo al tamaño de la fuente utilizada.

**X-height (ex):** Expresa el tamaño relativo al de la letra "x".

**Pixel (px):** Expresa el tamaño relativo a la resolución del monitor.

* *Absolutas:*

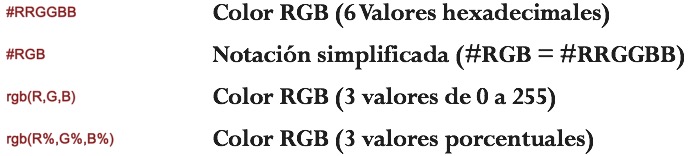
**Milímetros (mm)**.

**Centímetros (cm**):Cada centímetro son 10 mm.

**Pulgadas (in):** Cada pulgada equivale a 2,54 cm.

**Puntos (pt**): Cada punto son 1/72 in.

**Picas (pc**): Cada pica son 12 pt.

****

1. **Pseudoclases.**
   1. Las pseudo-clases son usadas para dar efectos especiales a algunas selectores:

*Selector:pseudo-clase {propiedades:valores;}*

* 1. O con clases:

*Selector.clase:pseudo-clase {propiedades:valores;}*

**6.1. :first-child.**

La pseudo-clase :first-child selecciona el primer elemento hijo de un elemento. Ejemplo:

*p em:first-child {*

*color: red;*

*}*

*<p>Lorem <span><em>ipsum dolor</em></span> sit amet, consectetuer adipiscing elit. Praesent odio sem, tempor quis, <em>auctor eu</em>, tempus at, enim. Praesent nulla ante, <em>ultricies</em> id, porttitor ut, pulvinar quis, dui.</p>*

El selector p em:first-child selecciona el primer elemento <em> que sea hijo de un elemento y que se encuentre dentro de un elemento <p>. Por tanto, en el ejemplo anterior sólo el primer <em> se ve de color rojo.

**6.2. :lang.**

La pseudo-clase :lang se emplea para seleccionar elementos en función de su idioma. Los navegadores utilizan los atributos lang, las etiquetas <meta> y la información de la respuesta del servidor para determinar el idioma de cada elemento. Ejemplo:

*p { color: blue; }*

*p:lang(es) { color: red; }*

Los párrafos del ejemplo anterior se ven de color azul, salvo los párrafos cuyo contenido esté escrito en español, que se ven de color rojo.

**6.3. :link.**

La pseudo-clase :link se aplica a todos los enlaces que todavía no han sido visitados por el usuario.

*a:link { color: red; }*

**6.4. :visted.**

La pseudo-clase :visited se aplica a todos los enlaces que han sido visitados al menos una vez por el usuario.

*a:visited { color: green; }*

1. **Pseudoelementos.**

Si se desea por ejemplo cambiar el estilo de la primera línea de texto de un elemento, la única forma de poder seleccionar estos elementos especiales es mediante los pseudo-elementos definidos por CSS para este propósito.

**7.1. :first-line.**

Permite seleccionar la primera línea de texto de un elemento. La siguiente regla CSS muestra en mayúsculas la primera línea de cada párrafo:

*p:first-line { text-transform: uppercase; }*

**7.2. :first-letter.**

Permite seleccionar la primera letra de la primera línea de texto de un elemento. De esta forma, la siguiente regla CSS muestra en mayúsculas la primera letra del texto de cada párrafo:

*p:first-letter { text-transform: uppercase; }*

**7.3. :before y :after.**

Se utilizan en combinación con la propiedad content de CSS para añadir contenidos antes o después del contenido original de un elemento.

Las siguientes reglas CSS añaden el texto Capítulo - delante de cada título de sección <h1> y el carácter . detrás de cada párrafo de la página:

*h1:before { content: "Capítulo - "; }*

*p:after { content: "."; }*

El contenido insertado mediante los pseudo-elementos :before y :after se tiene en cuenta en los otros pseudo-elementos :first-line y :first-letter

1. **Etiquetas <div>, <span>.**

Nos permite crear un estilo que queremos que actúe sólo, un estilo completo creado de la nada. Una etiqueta nueva y propia. Entonces utilizamos las etiqueta <SPAN> y <DIV>.

La diferencia entre ambas es que, mientras SPAN realmente no hace nada por sí misma, DIV convierte a todo lo que encierra en un bloque aparte (poniendo un salto de línea tanto al comienzo como al final).

1. **Comentarios.**

Para introducir un comentario en una pagina CSS se hace acotándola entre /\* comentario\*/

1. **Reglas @.**

* Regla @import

Agrega los estilos CSS de otra hoja en la que lo usa. Ejemplo:

*@import url(addonstyles.css);*

* Regla @media

Aplica las reglas que contiene al tipo de medio específico que indica (para impresiones, uso normal por pantalla, para proyectores, etc.). Ejemplo:

@media print {

body {

font-size: 10pt;

font-family: times new roman, times, serif;

}

#navigation {

display: none;

}

}

Los tipos de media que se pueden utilizar son:

all - Se aplica a todos los tipos.

aural - Para sintetizadores de voz.

handheld - Para dispositivos de mano (Palm, iPhone, etc).

print - Para impresoras.

projection - Para proyectores.

screen - Para pantallas de computadora.

* Regla @charset

Ajusta el tipo de codificación de caracteres basándose en los que utiliza otra hoja de estilo. Por ejemplo:

*@charset "ISO-8859-1";*

* Regla @page

Es utilizada para configuraciones avanzadas a la hora de imprimir un sitio, permite por ejemplo determinar el tamaño con el que se imprimirá o con qué margen. Por ejemplo:

*@page {*

*size: 15cm 20cm;*

*margin: 3cm;*

*marks: cross;*

*}*

**T4. XML.**

**\*Sólo preguntas del libro.**

1. **Introducción.**

**XML (eXtensible Markup Language)** es un formato de almacenamiento de información a base de arcas definidas por el usuario. Debe de cumplir una serie de reglas para que esté bien formado.

Existen dos técnicas para validar un documento XML, DTD, que es algo obsoleta pero está muy extendida, y esquemas XML que es más sofisticada con niveles de definición más precisos.

Algunos de los lenguajes basados en XML para usos específicos son, SVG, WML, RSS, FO.

Y otros formatos de almacenamiento alternativos a XML son JSON y YAML.

1. **Documentos XML.**

**XML (eXtensible Markup Language)** es un estándar, un metalenguaje de marcas y no dispone de un conjunto fijo de etiquetas. Por el contrario permite a los desarrolladores definir los elementos que necesiten y con la estructura que mejor les convenga.

**Lo que no es XML:**

* No es un lenguaje de programación. No existen compiladores de XML que generen ejecutables a partir de un documento XML.
* XML no es un protocolo de comunicación, no envía datos a través de internet (no como http (HyperText Transfer Protocol) o FTP (File Transfer Protocol).
* XML no es un sistema gestor de bases de datos. Una base de datos relacional puede contener campos del tipo XML. Pero XML en si mismo no es una base de datos.
* No es propietario, no pertenece a ninguna compañía como sucede con otros formatos.

1. **Espacios de nombre (Namespaces).**

Es un mecanismo para evitar conflictos de nombres, de manera que se puedan diferenciar elementos o atributos dentro de un mismo documento XML que tengan idénticos nombres pero diferentes definiciones.

Se declaran como atributo de elementos de la forma:

<nombre\_elemento xmlns:prefijo=”url”>

Ejemplo:

<cliente>

<nombre> Juan Pérez</nombre>

<mascota>

<nombre>Scooby</nombre>

</mascota>

</cliente>

<cli:cliente xmlns:cli=”www.prueba.com/cliente”>

<cli:nombre>Juan Perez</cli:cliente>

<pet:mascota xmlns:cli=”www.prueba.com/pet”>

<pet:nombre> Scooby</pet:nombre>

<pet:mascota>

</cli:cliente>

1. **Parser XML.**

Un analizador XML (parser), es un procedasodr que lee un documento CML y determina la estructura y propiedades de los datos en él contenidos. Lee el documento y genera un árbol jerárquico asociado, lo que permite ver los datos en un navegador o ser tratados por cualquier aplicación.

Si el analizador comprueba las reglas de buena formación y además valida el documento contra un DTD o esquema XML, es un analizador validador. También comprueban la semántica del documento e informan de los errores existentes.

[www.xmlvalidation.com](http://www.xmlvalidation.com)

1. **Documentos XML Válidos (apuntes XML)**
2. **DTD. Componentes (apuntes XML).**
3. **Lenguajes basados en XML.**

Existen lenguajes basados en XML que se usan para propósitos específicos. Estos lenguajes tienen su mecanismo de validación que fijará que elementos y atributos pueden aparecer.

* **SVG.** Scalable Vector Graphics. Es un lenguaje de representación de gráficos vectoriales bidimensionales. Utiliza etiquetas propias que describen elementos de dibujo como <canvas>, <shape> o <curve>. Se le aplica también una hoja de estilos externa de CSS.
* **WML.** Wireless Markup Language. Es un lenguaje de representación de la información que se visualiza en las pantallas de dispositivos móviles que utilicen el protocolo Wireless Application Protocol.
* **RSS.** Really Simple Syndication. Es una familia de formatos de semillas web usadas para publicar información que se actualiza con frecuencia.
* **Atom.** Atom Syndication Format. Es un lenguaje usado como semillas web, alternativa a RSS.
* **DocBook.**  Es un lenguaje de definición de documentos.
* **XBRL.** Extensible Bussines Reporting Language. Es un estándar abierto que permite representar la información y la expresión de la semántica requerida en los informes financieros.

1. **Lenguajes de almacenamiento de información.**

* **JSON.** JavaScript Object Notation. Es un formato en modo texto para el almacenamiento y transmisión de información. Se usa como alternativa a XML junto con AJAX (Asynchronous JavaScript And XML).

El procesado de un documento en formato JSON es más sencillo que el de uno en XML. En ocasiones se usan de manera conjunta ambas tecnologías. JSON se apoya en dos estructuras de datos, una colección de pares nombre/valor y una lista ordenada de calores.

* **YAML.** YAML Ain’t Markup Language. Es un formato de almacenamiento de información o socialización de datos, ligero y de fácil lectura para el ojo humano. Comparte con JSON ciertos elementos como las listas y los arrays asociativos.

**T5. Tratamiento y recuperación de datos.**

1. **XPath.**

XPath forma junto XSLT y XSL-FO una familia de lenguajes llamada XSL, diseñados para acceder, transformar y dar formato de salida a documentos XML.

Un documento XML tiene una estructura de un árbol, un elemento raíz que tiene hijos, que a su vez tienen otros hijos. A su vez, cada elemento puede tener atributos y/o contenido textual. Se asemeja a un árbol genealógico donde aparecerán conceptos como padre, hijo, hermano, ascendiente o descendiente.

Para recorrer y listar la estructura de directorios usaríamos los comandos adecuados del sistema operativo. Sin embargo en un documento XML para recorrerlo y extraer su información se usará Xpath.

Por lo tanto, XPath es un lenguaje de expresiones que no se usa de manera independiente, sino que se emplea en el contexto de un lenguaje anfitrión, como XSLT.

1. **Direccionamiento. (apuntes XPath).**
2. **XQuery.**

XQuery es un lenguaje de constula que permite extraer y procesar información almacenada en formato XML, habitualmente en bases de datos nativas XMl o en tablas y campos de tipo XML en bases de datos relacionales.

Se asemeja a SQL en algunas cláusulas empleadas y a XPath en el modelo de datos y funciones y operadores.

1. **Definiciones.**

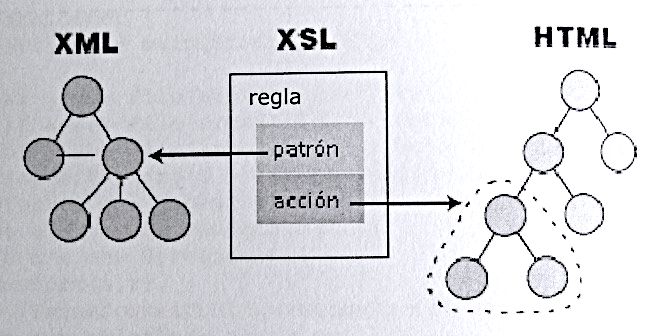
* **SAX.** Simple API for XML. Es una API usada para manejar XML en Java. Está orientada a la lectura de documentos XML. El analizador va leyendo el documento XML y enviando notificación de ello al programa en tiempo real. No es necesario leer el documento entero para poder trabajar sobre los datos que se encuentran al principio del mismo. Es elección idónea para documentos grandes que no caben en la memoria disponible.
* **DOM.** Document Object Model. Es una API para acceder y manipular documentos XML tratándolos como estructuras de árbol de nodos. Es un conjunto de recomendaciones que describen el modelo de objetos, independiente de cualquier lenguaje de programación, usado para almacenar documentos jerárquicos en memoria. Es una tecnología muy adecuada para aplicaciones que requieren acceder aleatoriamente a diferentes posiciones del árbol, pero no para documentos pesados.
* **JAXP.** Java API for XML Processing. Es una API que permite a las aplicaciones Java procesar, transformar, validar y consultar documentos XML.
* **XPOINTER.** XML Pointer Language. Es una especificación para vincular fragmentos de documentos XML. Se utiliza conjuntamente con XLink. Se apoya en XPath para identificar los fragmentos del documento XML que se enlazarán.
* **XLINK.** XML Linking Language. Es un estándar para vincular documentos XML. Se pueden vincular documentos XML mediante enlaces simples, equivalentes a la etiqueta <a> de HTML que permiten vincular un documento con otro, y mediante enlaces extendidos que permiten vincular varios documentos entre sí.
* **JAXB.** Java Architecture for XML Binding. Es una API alternativa a JAXP, para facilitar el manjeo de documentos XML así como optimizar el rendimiento. Requiere un esquema XSD asociado al XML.

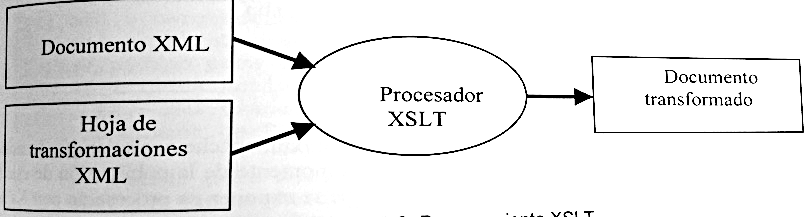
**T6. Transformación de documentos. XSLT.**

1. **XSLT.**

XSLT está basado en XML, cada hoja de transformaciones es un documento XML bien formado.

Es un lenguaje de programación declarativo que permite generar documentos en diferentes formatos de salida (XML, HTML, texto, PDF, RTF, etc…), a partir de un documento XML.

Consiste en la de declaración de una serie de reglas que hay que aplicar a un documento XML para transformarlos en la salida deseada. Estas reglas se almacenarán en un documento de texto, que habitualmente tiene la extensión .xsl, que junto con el documento .xml de partida serán pasados como parámetros a un procesador XSLT, que generará como salida el nuevo documento transformado.

****

1. **CSS vs XSLT.**

Existen dos familias de normas para hojas de estilo, la más antigua y simple es CSS, que permite definir propiedades de elementos de marcado, sin embargo hay ciertas tareas que **no se pueden** hacer con CSS:

* Cambiar el orden en el que los elementos de un documento HTML se visualizan.
* Realizar operaciones, como sumar valores de todos los elementos <precio> de un documento.
* Combinar múltiples elementos como las nóminas anuales de un empleado.

Estas limitaciones se superan con XSLT, en muchas ocasiones se usarán CSS y XSLT conjuntamente y de manera complementaria, el primero para dar formato visual y con el segundo determinaremos el contenido de un documento XML.

1. **Procesador XSLT.**

Son aplicaciones capaces de procesar documentos XML mediante hojas de transformaciones XSLT.

El procesador lee un documento XML, genera una representación de dicho documento como un árbol de nodos. Los tipos de nodo son, elemento, atributo, texto, comentario, instrucción de procesamiento y espacio de nombres. Las relaciones entre los nodos son las mismas que en un árbol genealógico, padres, hijos, ascendientes, descendientes….

El procesador XSLT va recorriendo y procesando nodo a nodo. El nodo que se está tratando en un momento dado se denomina nodo de contexto.

1. **Estructura básica de una hoja de transformaciones.**

Una hoja de transformaciones estará compuesta por:

1. Una declaración de documento XML: <?xml version=”1.0”>
2. Un elemento raíz, llamado <xsl:stylesheet>
3. El espacio de nombres, <http://www.w3.org/1999/XSL/Transform>, siendo el xsml el prefijo.

<?xml version=”1.0” encoding=”UTF-8”>

<xsl:stylesheet version=”1.0”

xmlns:xsl= “<http://www.w3.org/1999/XSL/Transform>”>

…

</xsl:stylesheet>

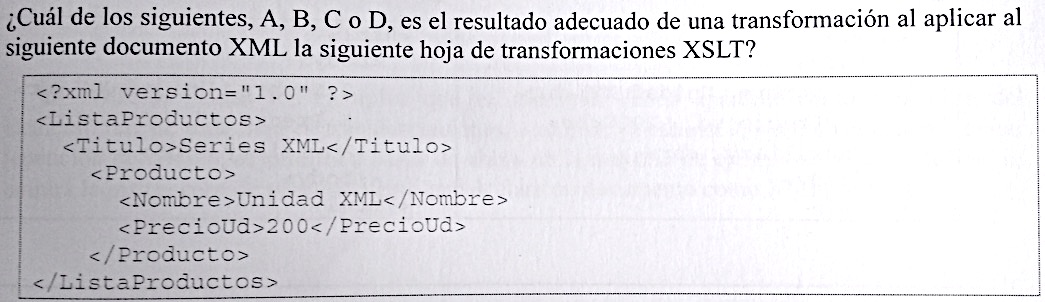
1. **Enlazar documentos XML con hojas de estilo.**

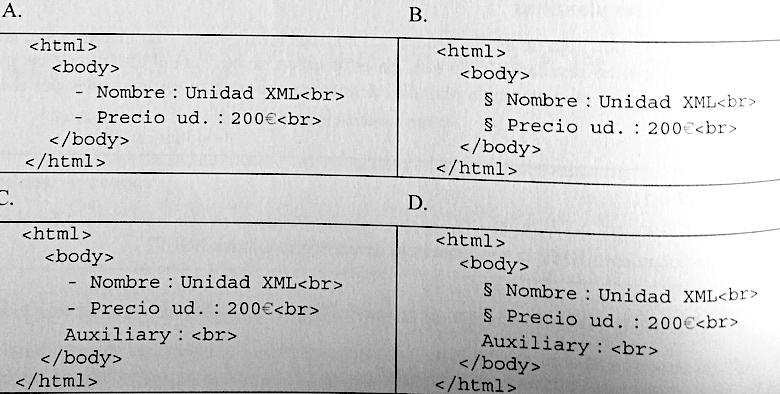
Se puede asociar de forma permanente una hoja de estilo, CSS o XSLT, a un documento XML mediante la siguiente instrucción, que se ubicará al principio de documento:

<?xml version=”1.0” encoding=”UTF-8”>

**<?xml-stylesheet href=”estilos.css”>**

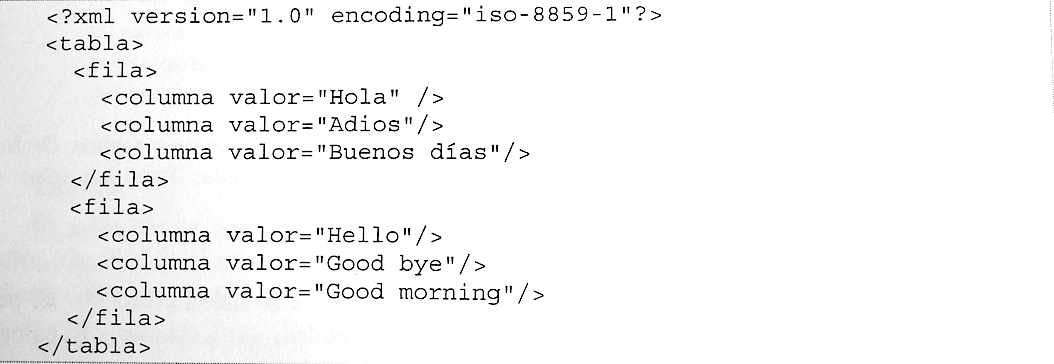
1. **Actividad página 319.**



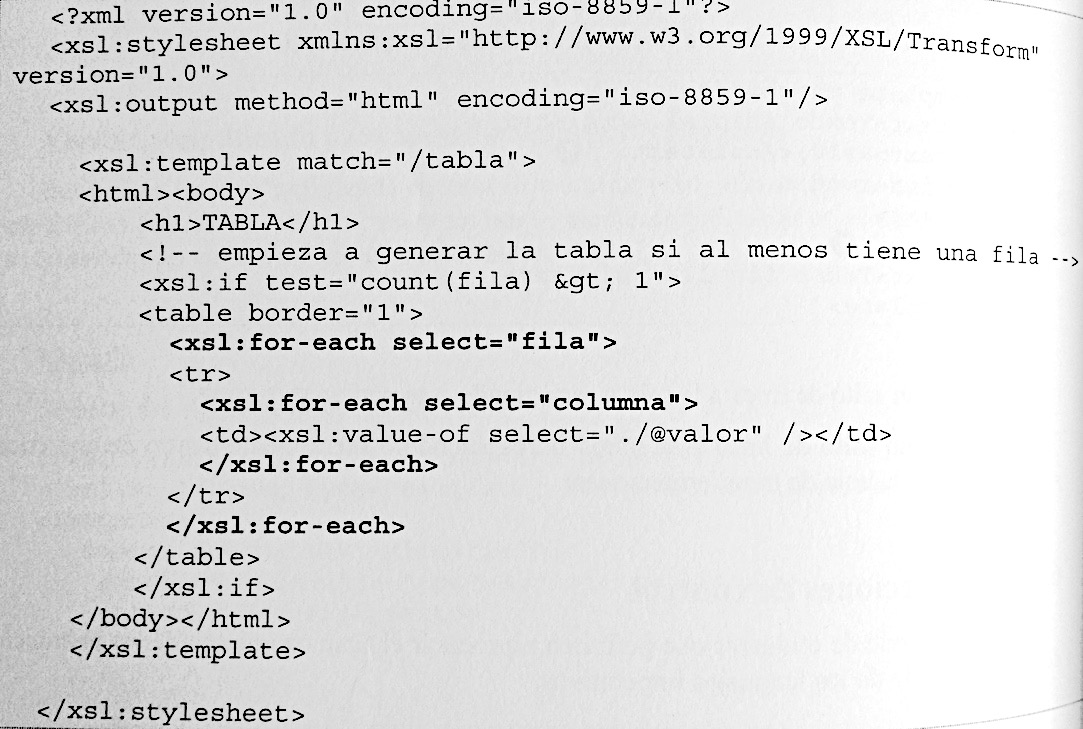
****

1. **Ejemplo página 326.**

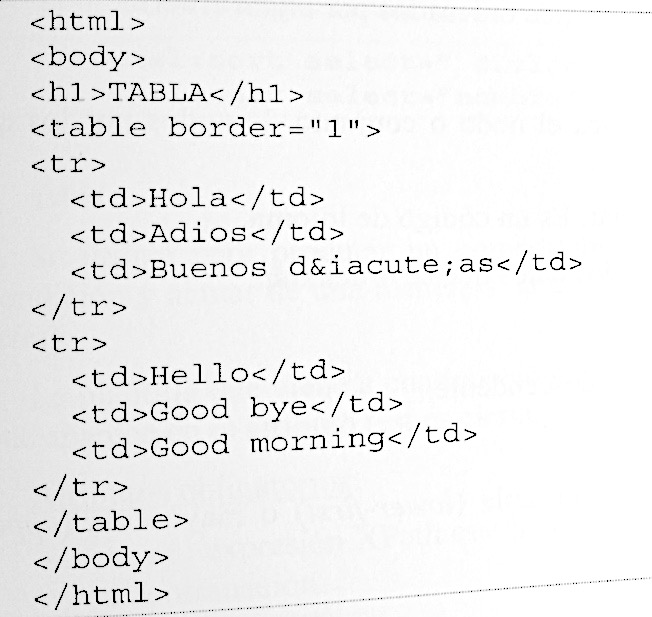
Se dispone de un documento XML de partida. Se va a convertir en documento HTML de salida que dibuje una tabla con las palabras mencionadas:



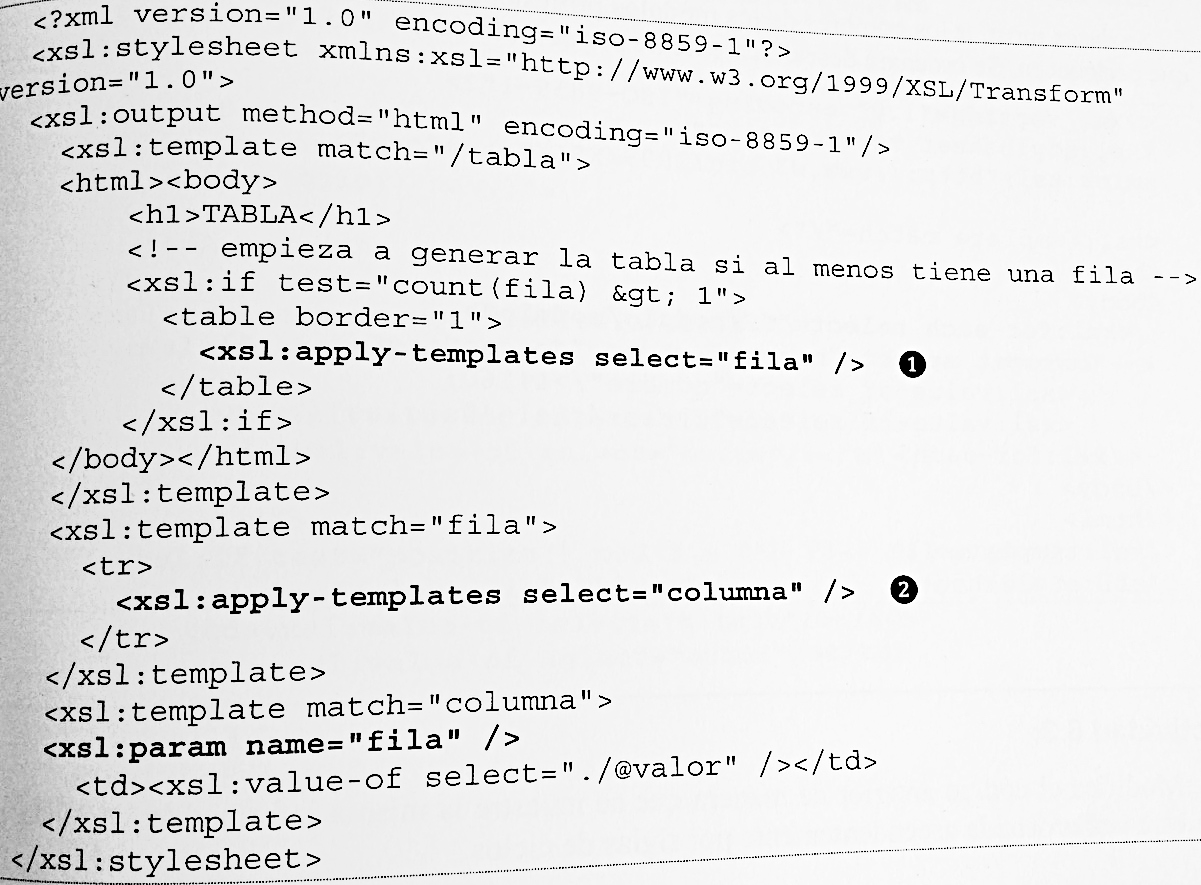
La hoja de transformación será:



La salida queda:

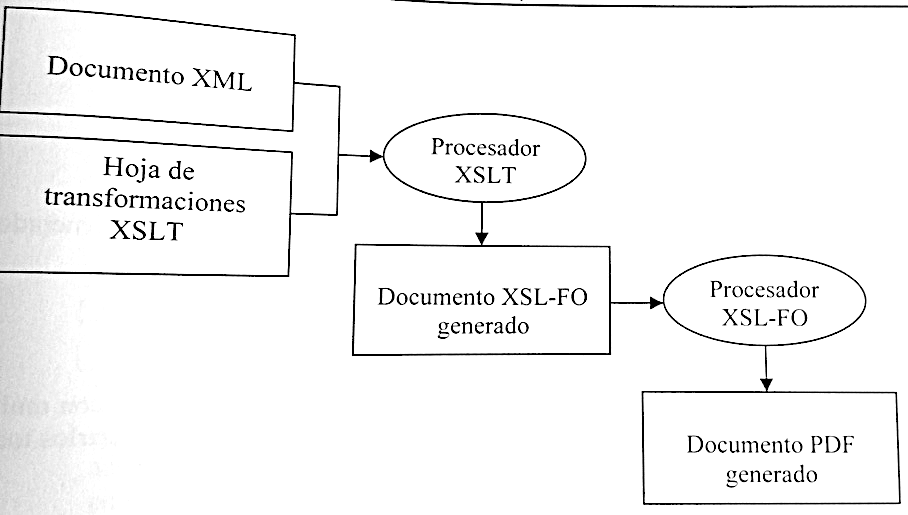


El mismo efecto queda con esta otra hoja de transformaciones, donde se crea una plantilla que se aplica para cada fila y otra para cada columna dentro de la fila:



1. Se invoca la plantilla para cada elemento /tabla/fila.
2. Se invoca la plantilla para cada elemento /tabla/fila/columna.
3. **XSL-FO.**

XSL-FO (XSL Formating Objects). Es un lenguaje de marcas que permite especificar el aspecto ue tendrán una serie de datos al mostrarlos en un determinado formato de salida, habitualmente un PDF (Portable Document Format) o RTF (Rich Text Format).



1. **FOP.**

FOP (Formating Object Processor), es una aplicación Java, dentro del proyecto Apache que toma como entrada un documento XSL-FO y genera una salida en algún formato visible como PDF.

**T7. Sindicación de contenidos. RSS.**

1. **RSS.**

RSS, Really Simple Syndication, es una tecnología que sirve para compartir o distribuir información en la web. Cuando un usuario está interesado en un tema, y quiere recibir información continuada, se suscribe a la fuente de contenidos de dicho tema. De esta forma el usuario no tiene que preocuparse de consultar periódicamente la fuente para ver si existe nueva información.

Es un lenguaje derivado de XML, aunque no es único existente para este propósito, también está Atom.

**Ventajas de la sindicación de contenidos:**

* Los usuarios no necesitan comprobar si la información ha sido actualizada.
* El formato de los datos es texto plano, ligero y rápido para transmitirse, ideal para dispositivos móviles.
* Se puede filtrar información que nos interesa de cada sitio y no perder tiempo con el resto de información.
* Protege la cuenta de correo puesto que no es necesario utilizarla.

1. **Estructura de un documento RSS.**
2. Declaración XML.

<?xml version=”1.0” encoding=”UTF-8”>

1. Declaración de tipo de documento.

<rss version=”2.0”>

1. Raíz del documento.

<rss> contenido </rss>

1. <channel> que describe la fuente o canal RSS, que a su vez tiene tres elementos obligatorios, <title>, <link> y <description>. El elemento channel puede tener uno o varios elementos <ítem> y cada elemento <ítem> define un artículo en el canal RSS.

<?xml version=”1.0” encoding=”UTF-8”>

<rss version=”2.0”>

<channel>

<title>Bienvenidos a mi pagina</title>

<link>http://www.ansango.com</link>

<description>ansango información</description>

<item>

<title>programación</title>

<link> http://www.ansango.com/rss/programacion.rss</link>

<description>contenido programación</description>

</item>

</channel>

</rss>

1. **Elementos de un RSS.**
   1. **<channel>.**

El elemento <channel> tiene tres elementos obligatorios:

* <title>, nombre del canal.
* <link>, hipervínculo para el canal.
* <description>, contenido del canal.

Elementos opcionales en <channel> (entre otros):

* <category>. Agrupar sitios basándose en la categoría.
* <copyright>. Derechos de autor.
* <image>. Imagen cuando los agregadores presentan un canal.
* <language>. Idioma utilizado para escribir el documento RSS.
* <pubDate>. Fecha de publicación.
* <rating>. Valoración PICS del canal.
  1. **<item>.**

Cada elemento <item> define un artículo en el canal RSS. Tiene tres elementos necesarios:

* <title>, nombre del artículo.
* <link>, hipervínculo al artículo.
* <description>, describe el artículo.

Elementos opcionales en <channel> (entre otros):

* <author>. Email del autor del artículo.
* <category>. Categoría del artículo.
* <comments>. Comentarios sobre el tema.
* <enclousure>. Inclusión de archivo multimedia..
* <guid>. Identificador único para el artículo.
* <source>. Especifica la fuente mediante un link.
* <pubDate>. Fecha de publicación.

**T8. Sistemas de gestión de información. ERP.**

1. **Introducción.**

Las herramientas BI (Business Intelligence), permiten registrar información sobre distintos aspectos de una determinada actividad. ERP suponen sistemas modulares integrados de gestión de empresas y CRM, es un tipo de aplicación destinada a la gestión de clientes.

1. **Inteligencia del negocio.**

Desde la aparición de los ordenadores personales, los individuos han sido capaces de registrar datos en sistemas de información más o menos sofisticados.

En el momento en el que existen gran cantidad de datos, fácilmente procesables y recuperables, se han podido estudiar los diferentes procesos de cualquier actividad. Esto permite aprender el funcionamiento de estos procesos, optimizarlos y transformarlos. Se ha generado por tanto inteligencia de negocio.

Las aplicaciones de inteligencia de negocio son aquellas que automatizan y agilizan procesos, registran resultados, favorecen la extracción de conclusiones…

Las familias de aplicaciones son las siguientes:

* **ERP**. Enterprise Resource Planning. Son aplicaciones integrales y modulares, para cubrir las necesidades de un negocio en casi todas las áreas o por paquetes.
* **CRM**. Customer Relationships Managment. Son aplicaciones enfocadas a la gestión de realciones con clientes.
* **CM**. Change Managment. Su fin es facilitar a los empleados los cambios estructurales que puedan tener lugar.
* **BA**. Business Analytics. Su objetivo es analizar datos, resultados, tendencias y sacar concluisiones.
* **ETL**. Extract, Transform and Load data. Fusión y homogeneización de datos.
* **SFA**. Sales Force Automation. Sistema de automatización de ventas.
* **BPM**. Business Process Managment. Gestión de procesos de negocio.
* **MRP**. Material Resource Planning. Planificación de recursos materiales.
* **SRM**. Supplier Relationship Manager. Relación con proveedores.

* **KM**. Knowledge Managment. Gestión del conocimiento.

1. **ERP.**

Durante el trascurso de la Segunda Guerra Mundial el gobierno de EEUU, desarrolla unos programas especializados para organizar las tropas y controlar la logística militar conocidos como Planificación de Requerimiento de Materiales (MRP). Estos sistemas empezaron a utilizarse en sistemas productivos en los años 50, los mismos fueron evolucionando con numerosos cambios tecnológicos y que la empresa Gartner bautizaría como ERP.

ERP. Enterprise Resource Planning. Es un sistema de gestión de información interna y externa de una empresa, incluyendo finanzas, contabilidad, ventas, compras, relaciones con clientes, almacenes.

Un componente estrella de ERP son los CRM dedicados a la gestión de contactos.

Tienen estructuras modulares y destacan por sus menús con base de datos central. Son aplicaciones integrales.

Los módulos principales son finanzas, control de gastos, logística, venta y distribución, gestión de materiales, ejecución logística, planificación de la producción, recursos humanos, componentes básicos, soluciones industriales, gestión de clientes, gestión de calidad.

Su **objetivo principal** es facilitar el flujo de información entre los diferentes estamentos de la empresa.

**Características:**

* Integral, cubre todas las necesidades.
* Modular, las aplicaciones se organizan por módulos independientes pero combinables.
* Parametrizable, con capacidad para adaptarse a necesidades concretas.
* Escalable, con capacidad de crecimiento.
* Acceso a base de datos centralizada.
* Captura datos automáticamente.
* Estructura modular.
* Configurables e interactivos.
* **Estructura básica:**
* *Infraestructura hardware*, determinada por los requisitos mínimos del fabricante, debe planificarse previamente.
* *Infraestructura software*, depende del hardware y posteriormente deberá seleccionarse la configuración adecuada.
* *Base de datos,*  dependerá del volumen que se desee almacenar.
* *La aplicación*, refleja los procesos internos de la empresa. El fabricante adapta las necesidades y funcionalidades que la empresa necesita. Integra los módulos de la empresa donde se gestionará la información.
* *La interfaz,* es la encargada de conectar los componentes anteriores y permitir así a los usuarios trabajar con la aplicación.
* **Interfaz del ERP.**
* *Cliente estándar*. Es una aplicación con toda la funcionalidad disponible que puede estar ejecutando las reglas de negocio en el mismo ordenador donde se ejecuta la propia aplicación. Ello requerirá una mayor necesidad de recursos.
* *Cliente ligero.* Es una aplicación diseñada para el mínimo de recursos. Por lo que no hay que instalar nada adicional y la tendencia es que sea a través del navegador web. Por tanto se facilita el acceso a dispositivos móviles.
* *Hoja de cálculo*. No suelen considerarse como parte del ERP, aunque un número de usuarios considerable presenta informes complejos mediante las mismas.
* *Cliente 100% web*. Se ejecuta desde el navegador.
  1. **SAP.** Sistemas, aplicaciones y productos.

Es una empresa alemana que desarrolla ERP comercial más utilizado. Esta compuesto de multitud de módulos que cubren la mayoría de los procesos de negocio que pueda necesitar cualquier empresa, orientados a finanzas, control de gastos, logística, venta y distribución, gestión de materiales, ejecución logística, planificación de la producción, recursos humanos, componentes básicos, soluciones industriales, gestión de clientes, gestión de calidad.

SAP integra un lenguaje de programación de cuarta generación propio, ABAP, que permite así mismo realizar las llamadas a procesos remotos para comunicarse con otros sistemas.

* 1. **OpenBravo.**

Es una alternativa a SAP en software libre. Se ejecuta a través de un cliente web y dispone de interfaz de fácil manejo.

Los módulos que se integran entre sí, pueden ser de gestión de datos, aprovisionamientos, almacenes, proyectos y servicios, producción, gestión comercial y CRM, finanzas y contabilidad, BI.

1. **CRM. SugarCRM.**

Customer Relationship Managment, es una aplicación informática orientada al registro de información de clientes, ventas y marketing. Su objetivo fundamental es ganar y mantener clientes.

Las tareas que suelen cubrir los CRM son, almacenado de datos, ofertas comerciales, informes y campañas publicitarias y de marketing.

Algunos de los CRM existentes son, Salesforce, CiviCRM, HiperGate, SugarCRM.

* **SugarCRM.**

Es un CRM de software libre. Los módulos más importantes son, ventas, marketing y servicio al cliente.

Las **características** de la arquitectura del servidor son; está desarrollado en LAMP, la lógica de negocio está implementada en PHP, tiene soporte para Windows, macOS y Unix, los sistemas gestores de bases de datos para los que hay versiones son SQL Server y Oracle.

La **aplicación puede ejecutarse** en la nube, de forma local y en dispositivos móviles.

Las **funciones principales**  son, mantenimiento de cuentas de empresas, de contactos de personas, de oportunidades de negocio, de documentos, de tareas, de notas y de empleados, creación de informes, de campañas publicitarias, registro de llamadas, envío de correo y planificación de reuniones.