Curso de HTML

Ing. Andrés Bustamante

Curso de HTML

Ing. Andrés Bustamante

fecha de publicación 26 de Marzo de 2009

Tabla de contenidos

Lista de figuras

Lista de tablas

Lista de ejemplos

Introducción

Este curso está en construcción permanente y hace parte de un material pedagógico para aprender a programar en HTML y luego en XHTML en 12 sesiones. Cada capítulo de este documento constituye una sesión, y cada sesión toma aproximadamente 3 horas, para aprender los conceptos explicados en el capítulo y practicarlos creando páginas Web con dichos conceptos.

Para el curso para el cual se desarrolla este material, se recomienda el uso del editor Amaya[1] como software de aprendizaje de HTML y XHTML, y en general para cualquier curso de HTML, ya que es una herramienta que permite combinar la teoría con la práctica de forma fácil y está comprobado que los estudiantes refuerzan lo aprendido utilizando esta herramienta de software, sin incurrir en costos por licenciamiento y sin requerir de un computador con grandes prestaciones para comenzar a crear páginas Web rápídamente.

Agradezco a Dave Raggett y al W3C (World Wide Web Consortium) para la realización de este documento, ya que varios de los ejemplos de este documento son adaptados de sus publicaciones en Internet, las cuales se referencian en la bibliografía al final de este documento.

[1] Amaya es desarrollado sin ánimo de lucro o interés comercial por el W3C (World Wide Web Consortium) y se puede descargar en la dirección en Internet http://www.w3.org/Amaya/.

Capítulo 1. Introducción a HTML

1.1. ¿Qué es HTML?

1.2. Historia de HTML

Capítulo 2. Aspectos básicos del lenguaje

En este capítulo se muestran los elementos básicos del lenguaje, en lo relacionado con la estructura de los documentos HTML, su sintaxis, y la forma en que podemos comenzar a elementos básicos a un documento HTML como lo son los el título, los encabezados, párrafos, imágenes, enlaces y listas. Adicionalmente, se revisa cómo darle formato simple al texto que se ingresa como parte del documento.

2.1. Estructura global de los documentos HTML

Todo documento en HTML debe componerse de tres partes:

1. Una línea con información de la versión de HTML que utiliza el documento. Corresponde a la definición del tipo de documento (DTD).

2. Una sección de encabezado de forma declarativa, con información adicional sobre el documento. Esta sección se delimita con el elemento `<head>`.

3. El cuerpo con el contenido del documento. Este cuerpo se delimita usualmente con el elemento `<body>`, aunque también se puede utilizar el elemento `<frameset>`, que veremos más adelante en el capítulo dedicado a marcos.

Las partes 2 y 3 se encuentran contenidas dentro del elemento principal de todo documento HTML, que es el elemento `<html>`. A continuación se presenta un ejemplo con la estructura global que debe tener cualquier documento HTML:

Ejemplo 2.1. Estructura global de los documentos HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
   <HEAD>
      <TITLE>Mi primer documento HTML</TITLE>
   </HEAD>
   <BODY>
      <P>Hola Mundo
   </BODY>
</HTML>
```

Los documentos HTML tienen la facilidad de que sus elementos y atributos pueden ser escritos en mayúscula o minúscula de forma indistinta, así como se permite en algunos casos que se deje una etiqueta de un elemento sin cerrar, como es el caso de la línea con el texto "Hola Mundo". Si se observa bien, se abre una etiqueta con el elemento `<P>` (párrafo), pero no se cierra. Un navegador Web puede permitir que esto ocurra y el documento puede ser interpretado y visualizado sin problema. Sin embargo, se recomienda escribir los nombres de los elementos únicamente en minúscula y no olvidar cerrar los elementos, es decir, incluir una etiqueta de inicio y una etiqueta de cierre por cada elemento, para facilitar la lectura del documento y encontrar posibles problemas fácilmente. De esta forma, el código anterior escrito "correctamente" quedaría de la siguiente manera:

Ejemplo 2.2. Estructura global de los documentos HTML, reescrito según la recomendación

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
      <title>Mi primer documento HTML</title>
   </head>
   <body>
      <p>Hola Mundo</p>
   </body>
</html>
```

2.2. Sintaxis básica

HTML es un lenguaje de marcado de documentos para el *World Wide Web*. Como se había explicado anteriormente, este marcado consiste en estructurar un documento con base en **etiquetas** con diferentes funciones. Como se observa en la sección anterior, las etiquetas pueden ser de inicio o de cierre. Las **etiquetas de inicio** constan de nombres de elementos y sus respectivos atributos, envueltos entre los signos < y >. Las **etiquetas de cierre** constan de un signo / que indica el cierre, y a continuación el nombre del elemento que se cierra, envueltos entre los signos < y >. El texto que se encuentra en medio de las etiquetas de inicio y de cierre es el **contenido del elemento**. Se puede observar esta diferenciación de forma gráfica con el ejemplo de la Figura 2.1, “Estructura de las etiquetas HTML (Fuente: Wikipedia)”.

Figura 2.1. Estructura de las etiquetas HTML (Fuente: Wikipedia)

Aparte de la estructura que tienen las etiquetas, pueden existir espacios y retornos de línea entre un elemento y otro, de forma que de la buena organización del documento depende su fácil lectura. En el ejemplo anterior para mostrar en pantalla una página con el texto "Hola Mundo", se observa que existen elementos contenedores de otros elementos, como el caso de los elementos `<html>`, `<head>` y `<body>`. Si hay elementos al interior de estos elementos contenedores, lo ideal es declarar estos elementos utilizando una sangría mayor en cada nivel de profundidad, para poder ver la organización del documento HTML más fácilmente al abrirlo con un editor de texto.

Se pueden incluir comentarios dentro del código HTML, utilizando una sola etiqueta con la siguiente estructura:

Ejemplo 2.3. Comentario en HTML

```
<!-- Esto es un comentario -->
```

Pueden existir una o varias líneas de texto dentro de la etiqueta, pero siempre se debe cerrar de la forma señalada.

2.3. Títulos

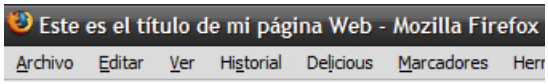
Los documentos HTML habitualmente llevan título. Este título es el texto que aparece en el navegador Web al momento de visitar la página, en la parte superior de la ventana, y la forma de incluirlo en el documento es utilizando el elemento `<title>` en la sección del encabezado del documento, como se muestra en el siguiente ejemplo:

Ejemplo 2.4. Agregando título a un documento

```
<head>
  <title> Este es el título de mi página Web </title>
</head>
```

El resultado en el navegador Web se observa en la Figura 2.2, “Ubicación habitual del texto del título de un documento HTML en un navegador Web”.

Figura 2.2. Ubicación habitual del texto del título de un documento HTML en un navegador Web



2.4. Encabezados y párrafos

Los encabezados o *headers* (por su nombre en inglés) son tipos de formato que se pueden aplicar sobre una porción de texto, para marcar divisiones de un documento a modo de secciones y subsecciones, de forma análoga a lo que se hace en documentos de texto elaborados con un procesador de texto como Microsoft Word u OpenOffice Writer.

En HTML, se permiten utilizar hasta 6 niveles de encabezamientos, y se declaran en el documento por medio de los elementos `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, y `<h6>`. No es requisito tener un encabezado `<h1>` para poder incluir un encabezado `<h2>` en el documento, y así sucesivamente, pero se recomienda que si se utilizan encabezados en el documento, se utilicen en la secuencia adecuada. Ésto le hace la vida más fácil a los buscadores o motores de búsqueda para indizar nuestros documentos en Internet, si es el caso, y facilita la lectura del documento para el usuario final.

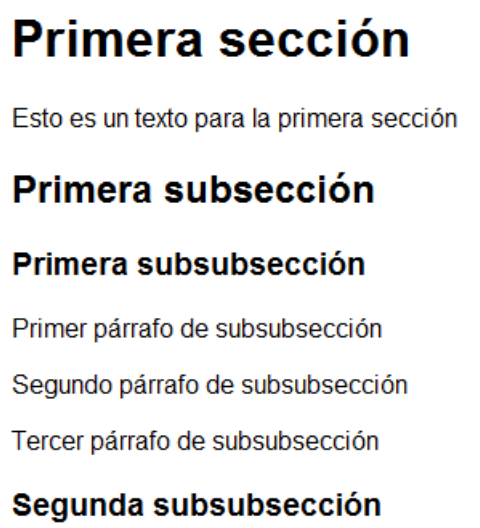
Los párrafos en HTML significan lo mismo que los párrafos gramaticalmente hablando. Los párrafos se inician por medio de la etiqueta `<p>`, y se cierran con la etiqueta de cierre respectiva, es decir, `</p>`. Sin embargo, los párrafos en HTML pueden quedar sin cerrar, de forma que el navegador Web cierra el párrafo automáticamente antes de comenzar uno nuevo, o antes de comenzar otro encabezado (sin importar el nivel). Veamos un ejemplo:

Ejemplo 2.5. Párrafos y encabezados de distintos niveles en HTML

```
<body>
  <h1>Primera sección</h1>
  <p>Esto es un texto para la primera sección</p>
    <h2>Primera subsección</h2>
      <h3>Primera subsubsección</h3>
      <p>Primer párrafo de subsubsección</p>
      <p>Segundo párrafo de subsubsección</p>
      <p>Tercer párrafo de subsubsección</p>
      <h3>Segunda subsubsección</h3>
</body>
```

El resultado de este fragmento de código HTML se observa en la Figura 2.3, “Visualización de encabezados de distintos niveles y párrafos para cada sección (encabezado)”.

Figura 2.3. Visualización de encabezados de distintos niveles y párrafos para cada sección (encabezado)



2.5. Imágenes

Una forma de hacer más agradables las páginas Web más allá del contenido de texto que pueda tener, es agregando imágenes en diferentes partes del cuerpo del documento. Estas imágenes se agregan en el documento utilizando el elemento `<img>` y varios de sus atributos, de acuerdo al caso. La imagen debe tener como mínimo el atributo `src` que hace relación a su ubicación con su respectivo valor. Esta ubicación puede ser una ruta completa a una imagen en Internet, o una ruta relativa al mismo dominio o subdominio donde se publica la página que se está editando. Observemos algunos ejemplos:

- Éste es un ejemplo con una ruta completa de imagen en Internet en alguna parte del contenido del documento:

Ejemplo 2.6. Agregando una imagen simple con una ruta completa

```
<img src="http://upload.wikimedia.org/HTML.svg" />
```

- Ahora un ejemplo con una imagen ubicada en el mismo subdominio de publicación, con una ruta relativa al mismo:

Ejemplo 2.7. Agregando una imagen simple con una ruta relativa

```
<img src="images/HTML.svg" />
```

- El mismo ejemplo, pero agregando atributos relacionados con el ancho y alto que va a tener la imagen al momento de mostrarse en el navegador:

Ejemplo 2.8. Agregando una imagen simple ancho y alto regulados

```
<img src="images/HTML.svg" width="200" height="150" />
```

- El mismo ejemplo, además agregándole un atributo especial para que muestre un texto específico en el caso de que por alguna razón no se logre mostrar la imagen:

Ejemplo 2.9. Agregando una imagen simple con ancho, alto y un texto alternativo

```
<img src="images/HTML.svg" width="200" height="150"
     alt="Imagen ilustrativa de HTML" />
```

Usualmente los navegadores Web interpretan formatos de imágenes comunes como GIF, JPEG y PNG, pero en la actualidad también soportan formatos de gráficos vectoriales, como SVG, que se utilizó en el ejemplo anterior. Cada formato tiene pros y contras, de forma que vale la pena recurrir a información detallada de estos formatos antes de incrustarlos como parte de los documentos HTML.

2.6. Enlaces

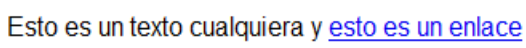
Es clave tener en HTML la facilidad de agregar a nuestros documentos enlaces a otras páginas. De hecho, eso es un factor de éxito para la WWW. Para definir un enlace con HTML utilizamos la etiqueta `<a>` y sus atributos de acuerdo al caso. El atributo de este elemento que habitualmente se utiliza es el atributo `href`, que indica hacia dónde apunta el enlace. Como contenido del elemento se escribe el texto que se va a ver en la página sobre el cual se puede accionar para activar el enlace y dirigir el usuario a la página señalada en el código. Este es un ejemplo:

Ejemplo 2.10. Enlace simple a una página del mismo sitio o subdominio

```
<p>Esto es un texto cualquiera y <a href="../enlace.html">esto es un enlace</a></p>
```

De nuevo, la ubicación del documento al cual se enlaza puede ser una ruta completa o una ruta relativa a ese documento, si el documento al que se apunta está en el mismo subdominio del documento en edición. En el caso anterior, tenemos una ruta relativa apuntando al documento `enlace.html` en el directorio superior al actual. El resultado del ejemplo anterior se observa en la Figura 2.4, “Visualización de un enlace simple”.

Figura 2.4. Visualización de un enlace simple



2.7. Listas

En HTML podemos tener listas sin orden específico (usualmente las conocemos como viñetas), listas con orden (numeradas) y listas de definición.

Las **listas sin orden** son listas donde todos los ítemes de la misma aparecen en pantalla marcados con una viñeta. Para definir esta lista y sus ítemes, utilizamos las etiquetas `<ul>` y `<li>` respectivamente, como se muestra en este ejemplo:

Ejemplo 2.11. Muestra de lista sin orden

```
<ul>
  <li>Primer ítem</li>
  <li>Segundo ítem</li>
  <li>Tercer ítem</li>
</ul>
```

Figura 2.5. Visualización de lista sin orden

- Primer ítem
- Segundo ítem
- Tercer ítem

Por otro lado, están las **listas ordenadas** donde no se muestran viñetas sino una numeración consecutiva en números arábigos. Este es un ejemplo:

Ejemplo 2.12. Muestra de lista ordenada

```
<ol>
  <li>Primer ítem</li>
  <li>Segundo ítem</li>
  <li>Tercer ítem</li>
</ol>
```

Figura 2.6. Visualización de lista ordenada

1. Primer ítem
2. Segundo ítem
3. Tercer ítem

El último tipo de lista de la cual se dispone en HTML es el de **listas de definición**, donde se tienen ítemes que corresponden a la estructura de términos con su respectiva definición, de forma que cada ítem es una pareja de elementos, así:

Ejemplo 2.13. Muestra de lista de definición

```
<dl>
  <dt>Primer término</dt>
  <dd>Su definición</dd>

  <dt>Segundo término</dt>
  <dd>Su definición</dd>

  <dt>Tercer término</dt>
  <dd>Su definición</dd>
</dl>
```

Esta lista se observaría de la forma que se presenta en la Figura 2.7, “Visualización de lista de definición simple”.

Figura 2.7. Visualización de lista de definición simple

Primer término
    Su definición
Segundo término
    Su definición
Tercer término
    Su definición

2.8. Resaltado simple de texto

Al igual que en los procesadores de texto, es posible señalar porciones de texto con formato de negrilla y cursiva para remarcar un texto. Sin embargo, existe una diferenciación en la forma de realizarlo, que puede ser por formato o por significado.

El **resaltado por formato** se realiza cuando únicamente nos interesa resaltar visualmente un texto, ya sea con negrilla o con cursiva. En este caso, utilizamos las etiquetas `<b>` e `<i>` para resaltar con negrilla o cursiva respectivamente.

Por otra parte, el **resaltado por significado** hace referencia a resaltar visualmente un texto y además hacer que su significado en el texto sea también resaltado, de forma que si se incluso si se tiene un lector de páginas Web para discapacitados, al pasar por este texto se resalte también, y no solo de forma visual. En este caso, las etiquetas a utilizar son `<strong>` y `<em>`. Aunque el resultado visual sea el mismo que con las etiquetas `<b>` e `<i>`, el resultado en el significado del texto es diferente.

Veamos un ejemplo:

Ejemplo 2.14. Resaltado de texto por formato y por significado

```
<p><strong>Esto es un texto</strong> <em>muy importante</em>, y
<b>este otro</b> <i>también</i></p>
```

La vista final de este fragmento de texto en el documento se observa en la figura Figura 2.8, “Visualización de resaltado simple por formato y resaltado por significado”.

Figura 2.8. Visualización de resaltado simple por formato y resaltado por significado

Esto es un texto *muy importante*, y **este otro** *también*

Capítulo 3. Aspectos avanzados del lenguaje (I)

Luego de ver los aspectos básicos de HTML, en este capítulo se presentan aspectos avanzados como el manejo de caracteres y símbolos especiales, el manejo de enlaces a partes específicas de un documento, y para terminar, una introducción al manejo de tablas en los documentos HTML.

3.1. Carácteres y símbolos especiales

En lo relacionado con los valores de los atributos y el contenido de los elementos, es posible introducir al momento de la edición cadenas de texto donde se utilicen cualquier clase de carácteres y símbolos de acuerdo a nuestro idioma y su correspondiente alfabeto. Sin embargo, al momento de la visualización del documento en un navegador Web pueden haber diferencias con respecto a lo que se ingresó, y pueden mostrarse carácteres que no corresponden a los esperados. Este es el caso de las vocales con tilde en español y la letra ñ, así como muchos otros carácteres especiales como el símbolo para el euro (la moneda), el símbolo de *copyright*, etc. Igual ocurre en otros idiomas, especialmente cuando no utilizan nuestro alfabeto.

HTML provee una facilidad para escribir estos carácteres por medio de **entidades**, de forma que los navegadores los muestren tal cual como queremos que aparezca. Estas entidades se manejan con códigos, que pueden ser escritos con letras o con números, como aparece en la siguiente tabla:

Tabla 3.1. Algunos ejemplos de caracteres especiales en HTML

Símbolo	**Entidad**
< (Menor que)	&lt;
> (Mayor que)	&gt;
& (Ampersand)	&amp;
" (Comillas)	&quot;
á	&aacute;
Á	&Aacute;
é	&eacute;
ñ	&ntilde;

En la especificación oficial de HTML se puede observar el detalle de las entidades válidas para HTML en la codificación Latin 1, que es la que habitualmente se usa en este lado del mundo para la visualización de documentos.[1]

Existen otros símbolos que habitualmente necesitamos usar en los documentos HTML, como los retornos de línea. Dado que la sintaxis de HTML permite retornos de línea en el código sin alterar el resultado, especialmente cuando se trata del contenido de los elementos, es necesario escribir en el código que explícitamente los vamos a mostrar. La forma de forzar un **retorno de línea** es utilizando la etiqueta `<br>`, que habitualmente no necesita etiqueta de cierre.

Por otro lado, a veces necesitamos que dos palabras funcionen como una sola, a pesar de que tienen espacio, como la referencia a una figura (Figura 1), o una marca (Coca Cola), y no queremos que se separen de línea las dos palabras sea cual sea la razón. En ese caso hay que reemplazar el espacio por una entidad HTML especial, que es la entidad de **espacio de no-rompimiento** ` `. En todo caso se recomienda no abusar de la utilización de esta entidad, especialmente para dar formato al documento, lo cual suele ser un vicio frecuente iniciando con HTML.

[1] La dirección del apartado con los caracteres especiales es http://www.w3.org/TR/html4/sgml/entities.html. También se puede obtener una guía visual de caracteres especiales en HTML y XHTML en la dirección http://www.digitalmediaminute.com/reference/entity/index.php.

Ejemplo 3.1. Utilizando retornos de línea y espacios de no-rompimiento en el contenido de los elementos

```
<p>Pedro Perez<br>
Director del Departamento de Ventas<br>
Mi Compa&ntilde;&iacute;a, Sede Bogot&aacute;<br>
2009<p>
```

3.2. Enlaces a partes de un documento

Los documentos en HTML pueden volverse muy extensos, y a veces se hace necesario hacer enlaces a algunas partes específicas de ese documento, especialmente los encabezados, para no enviar al usuario al inicio del documento y que tenga que buscar por sí mismo lo que queremos que vea. En este caso utilizamos la etiqueta `<a>`, que ya habíamos utilizado para generar los enlaces previamente, pero también sirve para generar las referencias al interior del documento. Para este fin ya no utilizamos el atributo `href`, sino el atributo `name`. El valor de este atributo debe ser único entre las demás referencias del documento, para poderlo enlazar fácilmente. Por ejemplo: tenemos el encabezado de nivel 2 "Cursos ofrecidos" y necesitamos un enlace a este título en el documento:

Ejemplo 3.2. Creación de una referencia al interior del documento

```
<h2><a name="cursos">Cursos ofrecidos</a></h2>
```

Para enlazar a este encabezado desde otra parte del mismo documento, utilizamos la etiqueta `<a>` para un enlace normal, pero el valor de `href` es el mismo valor de la referencia del título, antecedido por el símbolo `#`, así:

Ejemplo 3.3. Enlace a una parte específica referenciada del mismo documento

```
Visite nuestra secci&oacute;n de <a href="#cursos">cursos ofrecidos</a>.
```

De la misma manera, se puede enlazar este encabezado desde otro documento. El valor del atributo `href` tendría la ruta al documento que queremos enlazar, y luego la referencia al encabezado, así:

Ejemplo 3.4. Enlace a una parte específica de otro documento HTML

```
Visite nuestra secci&oacute;n de <a href="/inicio.html#cursos">cursos ofrecidos</a>.
```

3.3. Tablas

Las tablas son formas de organizar la información en un documento y constan de una o más **filas** con dicha información. Cada fila se divide en **celdas**, y estas celdas pueden tener o no tener tamaño definido, así como la tabla en sí.

Para definir una tabla, se utiliza la etiqueta `<table>` a modo de contenedor, y para definir cada una de sus filas, utilizamos la etiqueta `<tr>`. Cada una de las celdas en las filas de la tabla se enmarca con la etiqueta `<td>`, y se escribe `<th>` en vez de `<td>` cuando la celda es un **encabezado de la tabla**. Una tabla simple se escribe como en el ejemplo que sigue:

Ejemplo 3.5. Tabla simple con encabezado y datos

```
<table border="1">
  <tr><th>Nombre completo</th><th>Edad</th></tr>
  <tr><td>Pedro Perez</td><td>25</td></tr>
  <tr><td>Marta Sanchez</td><td>24</td></tr>
  <tr><td>Juan G&oacute;mez</td><td>27</td></tr>
</table>
```

El resultado de este fragmento de código se observa en la Figura 3.1, “Visualización de tabla simple con encabezado y datos”.

Figura 3.1. Visualización de tabla simple con encabezado y datos

Nombre completo	Edad
Pedro Perez	25
Marta Sanchez	24
Juan Gómez	27

3.3.1. Tablas con formato

Para ajustar el ancho de la tabla en el documento, se puede utilizar el atributo `width` de la etiqueta `<table>`. El valor puede ser dado en píxeles o en porcentaje. El porcentaje corresponde a la proporción de espacio entre la margen izquierda y la derecha del documento a la hora de ser visualizado.

Ejemplo 3.6. Tabla simple con encabezado y datos e indicación del ancho de la tabla

```
<table border="1" width="50%">
  <tr><th>Nombre completo</th><th>Edad</th></tr>
  <tr><td>Pedro Perez</td><td>25</td></tr>
  <tr><td>Marta Sanchez</td><td>24</td></tr>
  <tr><td>Juan G&oacute;mez</td><td>27</td></tr>
</table>
```

Figura 3.2. Visualización de tabla simple con indicación de ancho de la tabla por porcentaje

Nombre completo	Edad
Pedro Perez	25
Marta Sanchez	24
Juan Gómez	27

Se puede aumentar el **espaciado entre el texto y el borde de cada celda** utilizando el atributo `cellpadding` de la etiqueta `<table>`. El valor asignado para este atributo, al igual que el valor del borde está en píxeles.

Ejemplo 3.7. Tabla simple con encabezado y datos, con indicación de espaciado de texto al margen

```
<table border="1" cellpadding="10">
  <tr><th>Nombre completo</th><th>Edad</th></tr>
  <tr><td>Pedro Perez</td><td>25</td></tr>
  <tr><td>Marta Sanchez</td><td>24</td></tr>
  <tr><td>Juan G&oacute;mez</td><td>27</td></tr>
</table>
```

El resultado de este código de ejemplo es el de la Figura 3.3, “Visualización de tabla simple con encabezado y datos e indicación de espaciado entre el texto y el borde de las celdas”.

Figura 3.3. Visualización de tabla simple con encabezado y datos e indicación de espaciado entre el texto y el borde de las celdas

Nombre completo	Edad
Pedro Perez	25
Marta Sanchez	24
Juan Gómez	27

También se puede jugar con el **espacio entre una celda y otra** por medio del atributo `cellspacing` de la misma etiqueta `<table>`.

Ejemplo 3.8. Tabla simple con encabezado y datos, con indicaciones de espaciado de texto al margen y espaciado entre celdas

```
<table border="1" cellpadding="10" cellspacing="5">
  <tr><th>Nombre completo</th><th>Edad</th></tr>
  <tr><td>Pedro Perez</td><td>25</td></tr>
  <tr><td>Marta Sanchez</td><td>24</td></tr>
  <tr><td>Juan G&oacute;mez</td><td>27</td></tr>
</table>
```

El resultado se observa en la Figura 3.4, “Visualización de tabla simple con encabezado y datos e indicación de espaciado entre el texto y borde de las celdas, además de espacio entre celdas”.

Figura 3.4. Visualización de tabla simple con encabezado y datos e indicación de espaciado entre el texto y borde de las celdas, además de espacio entre celdas

Nombre completo	Edad
Pedro Perez	25
Marta Sanchez	24
Juan Gómez	27

Para tener una tabla sin bordes, se le asigna el valor de 0 a los atributos `cellspacing` y `border`. Esto es útil cuando se utiliza una tabla como organización de la información de todo el contenido del documento y no queremos que se vean las divisiones de la tabla.

Por otra parte, se puede cambiar la **alineación de los textos** de las celdas de forma diferente a como se hace por defecto por parte de los navegadores Web, es decir, los títulos centrados y el contenido normal a la izquierda. Este cambio se logra con el atributo `align` y su valor asociado en inglés (centro = *center*, izquierda = *left*, derecha = *right*). Este atributo lo tienen las etiquetas `<tr>` y `<td>`. En el caso de que la alineación aplique para toda la fila, se pone este atributo a la etiqueta `<tr>` respectiva, mientras que si la alineación aplica solo para una celda, se coloca el atributo align a la etiqueta `<td>` señalada.

También se pueden dibujar **celdas combinadas**, es decir, celdas que tomen más de una columna o de una fila, ya sea que se quieran combinar celdas horizontal o verticalmente, según sea el caso. Para lograr esto, se le agregan los atributos `colspan` (combinación horizontal) o `rowspan` (combinación vertical) a las etiquetas de datos de la celda, es decir a las etiquetas `<th>` o `<td>` según sea el caso. Los valores de estos atributos son numéricos, e indican la cantidad de espacios a combinar hacia la derecha (en el caso horizontal) o hacia abajo (en el caso vertical). Veamos esto aplicado en un ejemplo:

Ejemplo 3.9. Tabla avanzada con aplicación de alineación de textos y celdas combinadas

```
<table border="1" width="50%">
  <tr align="center"><th rowspan="2">Nombre</th><th colspan="3">Notas</th></tr>
  <tr align="center"><th>Nota 1</th><th>Nota 2</th><th>Nota 3</th></tr>
  <tr align="center"><td>Pedro P&eacute;rez</td><td>4.5</td><td>3.8</td><td>4.2</td></tr>
  <tr align="center"><td>Juan G&oacute;mez</td><td>2.5</td><td>4.5</td><td>4.3</td></tr>
</table>
```

Esta tabla se vería en un navegador como aparece en la Figura 3.5, “Visualización de tabla avanzada con alineación de texto y combinación de celdas”.

Figura 3.5. Visualización de tabla avanzada con alineación de texto y combinación de celdas

Nombre	Notas		
	Nota 1	Nota 2	Nota 3
Pedro Pérez	4.5	3.8	4.2
Juan Gómez	2.5	4.5	4.3

3.3.2. Tablas accesibles

Teniendo en cuenta que la navegación en Internet también es posible para personas con problemas de la vista, entre otros, se puede agregar información a las tablas para ser interpretadas por estas personas más fácilmente, aunque en algunos casos también aplique para personas sin problemas visuales.

Por ejemplo, se puede agregar un título a la tabla, como los que aparecen en las tablas de este documento, diciendo básicamente de qué se trata la tabla. Esto se logra introduciendo una etiqueta `<caption>` al interior de la tabla, justo antes de comenzar a describir las filas y sus celdas. Por defecto este título aparecerá en la parte superior de la tabla, como aparece en el ejemplo a continuación. Para colocar este título en la parte inferior, se utiliza el atributo `align="bottom"` al interior de la etiqueta `<caption>`.

Ejemplo 3.10. Tabla simple con título

```
<table border="1" width="50%">
  <caption>Lista de estudiantes de la clase</caption>
  <tr><th>Nombre completo</th><th>Edad</th></tr>
  <tr><td>Pedro Perez</td><td>25</td></tr>
  <tr><td>Marta Sanchez</td><td>24</td></tr>
  <tr><td>Juan G&oacute;mez</td><td>27</td></tr>
</table>
```

Figura 3.6. Visualización de tabla con título

Lista de estudiantes de la clase

Nombre completo	Edad
Pedro Perez	25
Marta Sanchez	24
Juan Gómez	27

Además del título, se puede agregar un **resumen** a la tabla, que no se visualiza, pero sí es leído en los navegadores para personas con problemas de ceguera. Este resumen se incluye como valor del atributo `summary` en la etiqueta `<table>`.

Ejemplo 3.11. Tabla simple con título y resumen

```
<table border="1" width="50%"
    summary="Información de los estudiantes: nombre, edad">
  <caption>Lista de estudiantes de la clase</caption>
  <tr><th>Nombre completo</th><th>Edad</th></tr>
  <tr><td>Pedro Perez</td><td>25</td></tr>
  <tr><td>Marta Sanchez</td><td>24</td></tr>
  <tr><td>Juan G&oacute;mez</td><td>27</td></tr>
</table>
```

Asimismo, se puede agregar información adicional a las celdas, especialmente cuando son encabezados en columnas o filas. La información que se introduce le dice a navegadores especiales que dicha celda es el encabezado para la fila o para la columna según el caso. A esto se le denomina **alcance de la celda**, y se describe en el código con el atributo `scope` de las celdas, es decir, las etiquetas `<th>` o `<td>`, según el caso.

Ejemplo 3.12. Tabla simple con título, resumen y alcance de celdas de encabezado

```
<table border="1" width="50%"
    summary="Información de los estudiantes: nombre, edad">
  <caption>Lista de estudiantes de la clase</caption>
  <tr><th scope="col">Nombre completo</th><th scope="col">Edad</th></tr>
  <tr><td>Pedro Perez</td><td>25</td></tr>
  <tr><td>Marta Sanchez</td><td>24</td></tr>
  <tr><td>Juan G&oacute;mez</td><td>27</td></tr>
</table>
```

Capítulo 4. Aspectos avanzados del lenguaje (II)

Los temas que se tratan en este capítulo están relacionados con la inclusión de recursos multimedia en los documentos HTML. Estos recursos pueden ser archivos de audio (canciones, grabaciones, *podcasts*, etc.) y archivos de video (filmaciones, videos musicales, etc.); también se trata sobre la inclusión de *scripts* para la animación de las páginas por medio de *banners* o propaganda, y por medio de la animación de imágenes del documento HTML.

4.1. Agregando recursos multimedia al documento

Se pueden agregar recursos multimedia a los documentos HTML como archivos de sonido y video.

4.1.1. Agregando sonido

Suponga que tiene un archivo de audio en formato MP3 y lo ha subido a una carpeta de su servidor Web y quiere incluirlo en su página Web para que lo escuchen las personas que la visiten. Suponga que este archivo tiene la ruta http://example.com/music/myband.mp3. La forma más simple de hacer disponible este archivo de forma directa es crear un enlace a esta ruta en el documento, así:

Ejemplo 4.1. Enlace a archivo de audio en MP3

```
Esta es <a href="http://example.com/music/myband.mp3"
   type="audio/mpeg">mi canci&oacute;n favorita</a>.
```

Sin embargo, es mejor crear una lista de reproducción en formato M3U (un archivo de texto con las rutas de las canciones). Esto facilita que se comience a reproducir la canción sin esperar a que se descargue completamente el archivo con el audio. En este caso, luego de tener el archivo M3U se puede hacer el enlace así:

Ejemplo 4.2. Enlace a archivo de audio en MP3 utilizando una lista de reproducción

```
Esta es <a href="http://example.com/music/myband.m3u"
   type="audio/x-mpegurl">mi canci&oacute;n favorita</a>.
```

Asimismo se pueden utilizar archivos WAV, MIDI, AAC, OGG y otros formatos de audio utilizando un enlace directo o una lista de reproducción. Dependiendo del navegador y del sistema operativo del computador, se abrirá un reproductor adecuado para el tipo de formato utilizado.

También se pueden cambiar las rutas si los archivos se encuentran en el mismo servidor de nuestra página Web. Por ejemplo, la lista de reproducción con el archivo de audio puede ser enlazada así: `href="audio/myband.m3u"`.

Es importante utilizar el atributo `type` del enlace con el valor adecuado de acuerdo al tipo de archivo utilizado. Este valor corresponde al tipo MIME del recurso. Para el caso de archivos MP3, el tipo MIME es audio/mpeg, mientras que para la lista de reproducción en formato M3U el tipo MIME es audio/x-mpegurl.[1]

4.1.2. Agregando video

Es posible agregar videos a una página Web en HTML. Se pueden agregar de la misma manera que se agregaban archivos de audio al documento: a modo de enlaces. En este orden de ideas, primero hay que tener el video en formato AVI, MPEG, MP4, WMV, 3GP u otro formato propio de video, disponible desde un servidor Web que permita encontrar el video directamente, ya sea utilizando una ruta completa (en el caso de no encontrarse en el mismo subdominio de la página), o una ruta relativa (si el archivo está en el mismo subdominio de la página).

En el caso de los videos, no hay listas de reproducción, de forma que se enlaza a los videos directamente, así:

[1] Se puede consultar un listado más completo de tipos MIME en la dirección en Internet http://www.utoronto.ca/webdocs/HTMLdocs/Book/Book-3ed/appb/mimetype.html.

Ejemplo 4.3. Enlace a archivo de video en formato MPEG

```
Esta es <a href="http://example.com/video/myvideo.mpeg"
   type="video/mpeg">mi video favorito</a>.
```

Hay que fijarse de nuevo en la ruta del video y en el tipo MIME del archivo con el video. En el ejemplo anterior, el video está en formato MPEG, por tanto, el tipo MIME es video/mpeg.

Hoy en día, los videos pueden ser demasiado pesados para subir en un servidor Web y luego para descargarlos y visualizarlos en el navegador de nuestros visitantes, de forma que hay servicios especializados que permiten publicar nuestros videos en un formato más accesible y sencillo, de forma que no se tarde tanto comenzar la reproducción de video. Estos servicios son por ejemplo YouTube, Vimeo, DailyMotion, Metacafe y otros.

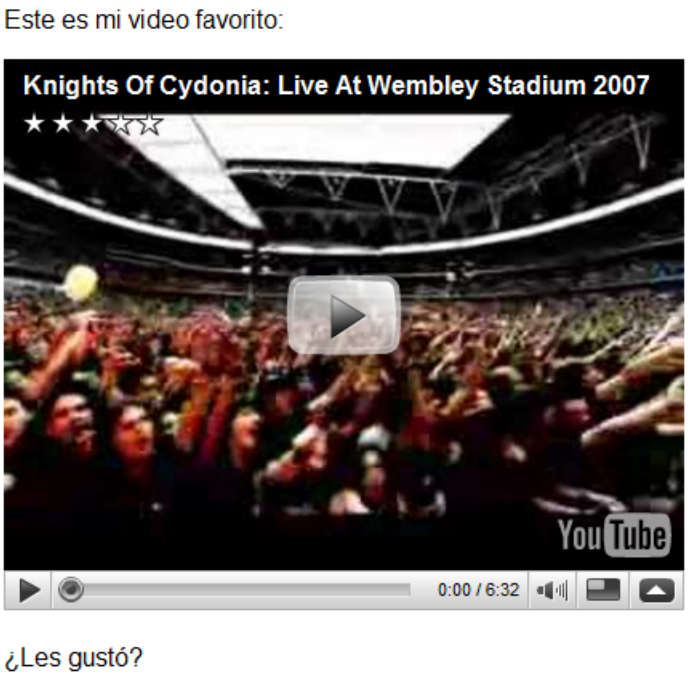
Si se tiene disponible una cuenta con alguno de estos servicios, es posible ver estos videos públicamente, y mejor aún, es posible utilizar incluir un fragmento de la página como nuestro video en ese servicio. Por ejemplo, en YouTube, al visualizar un video se muestra información para incluirlo en una página Web en HTML, copiando un código en HTML como este:

Ejemplo 4.4. Fragmento de código HTML para incluir un video de YouTube

```
<object width="425" height="344">
<param name="movie" value="http://www.youtube.com/v/j8WP7aOD_9Q&hl=en&fs=1"></param>
<param name="allowFullScreen" value="true"></param>
<param name="allowscriptaccess" value="always"></param>
<embed src="http://www.youtube.com/v/j8WP7aOD_9Q&hl=en&fs=1"
      type="application/x-shockwave-flash" allowscriptaccess="always"
      allowfullscreen="true" width="425" height="344">
</embed>
</object>
```

Este fragmento de código se vería como aparece en la Figura 4.1, “Video insertado desde YouTube en un documento HTML” en un navegador Web. Básicamente se inserta en la página un pequeño reproductor en Adobe Flash para el video ya existente en el servicio de YouTube (puede ser otro servicio), no es el video original como tal.

Figura 4.1. Video insertado desde YouTube en un documento HTML



4.2. *Scripting* simple

Además del código estático que se especifica en nuestros documentos HTML, se pueden agregar *scripts* para que los documentos sean más animados y llamen la atención en puntos específicos como imágenes, enlaces y otros,

de acuerdo a nuestras necesidades. Estos *scripts* son fragmentos de código de alguno de los siguientes lenguajes de programación:

- JavaScript
- Visual Basic Script
- TCL

La forma de incluir estos scripts en el documento HTML es utilizando el elemento `<script>`, especificando el tipo de script (el lenguaje) en el atributo `type`, y dejando en el contenido del elemento el código del *script*, así:

Ejemplo 4.5. Estructura básica de un script de JavaScript insertado en un documento HTML

```
<script type="text/javascript">
  // ... Código JavaScript ...
</script>
```

4.2.1. Animaciones simples

Es común querer animar imágenes que agregamos a nuestro documento HTML para llamar la atención. Usualmente se cambia la apariencia de las imágenes utilizando algún efecto como una sombra, cambiando el color de fondo o las posiciones de los objetos en la imagen. Veamos un ejemplo:

Ejemplo 4.6. Animación en JavaScript para cambiar una imagen-enlace mientras se pasa el cursor sobre ella (Original de Dave Raggett)

```
<script type="text/javascript">
if (document.images) {
    imagen1 = new Image;
    imagen2 = new Image;
    imagen1.src = "../images/enter1.gif";
    imagen2.src = "../images/enter2.gif";
}

function cambiarImagen(nombre, imagen) {
    if (document.images) {
        document[nombre].src = eval(imagen + ".src");
    }
}
</script>

...

<a href="#" onMouseOver='javascript:cambiarImagen("enter", "imagen2")'
    onMouseOut='javascript:cambiarImagen("enter", "imagen1")'>
  <img name="enter" src="../images/enter1.gif" alt="Enter if you dare!" />
</a>
```

Existen diferentes clases de eventos que se pueden utilizar para animar los objetos (imágenes especialmente) en nuestros documentos. Estos eventos están presentes como atributos de las etiquetas que permiten ejecutar *scripts* como el del ejemplo anterior. Para este caso solamente se estudiaron los eventos de ubicar el cursor del ratón sobre el enlace (la imagen) y retirarlo luego.

El resultado del ejemplo anterior se puede comprobar en este enlace.

4.2.2. Banners o avisos

Los *banners* en Internet hacen referencia a imágenes al interior de las páginas HTML relacionadas con patrocinadores, páginas "amigas" o "hermanas", colocadas de tal forma que llamen la atención de los visitantes y al hacer clic sobre ellos sean dirigidos hacia dicho patrocinador. Estos banners se pueden ubicar como avisos dinámicos,

de forma que utilizando una misma ubicación en la página, se muestren diferentes imágenes, y con cada imagen se enlace a sitios diferentes. A continuación se presenta un ejemplo con 3 imágenes diferentes y 3 enlaces asociados:

Ejemplo 4.7. Animación en JavaScript para mostrar avisos diferentes para enlaces diferentes (Original de Dave Raggett)

```
<html>
<head>
<title>Ejemplo</title>
<script type="text/javascript">
if (document.images) {
    imagenes = new Array("../images/batman.jpg",
                "../images/superman.jpg", "../images/flash.jpg");
    direcciones = new Array("www.batman.com",
                "www.superman.com", "www.flash.com");
    cuenta = 0;
}

function rotarAviso() {
    if (document.images) {
        if (document.mi_aviso.complete) {
            if (++cuenta == imagenes.length)
                cuenta = 0;

            document.mi_aviso.src = imagenes[cuenta];
        }
    }

    // Cambiar el aviso cada 3 segundos
    setTimeout("rotarAviso()", 3000);
}

function enlazarConAviso() {
    document.location.href = "http://" + direcciones[cuenta];
}
</script>
</head>

<body onload="rotarAviso()">

...

<a href="javascript:enlazarConAviso()">
  <img name="mi_aviso" src="../images/batman.jpg" border="0"
       alt="Mis superh&eacute;roes favoritos">
</a>
</body>
</html>
```

El resultado de este ejemplo se puede observar en este enlace. En el capítulo 8 se estará evaluando la forma de crear scripts con JavaScript con más profundidad.

Capítulo 5. Diseño con estilo

En este capítulo se presenta lo relacionado con agregarle algo de estilo a los documentos HTML. Para tal fin, se explicarán las formas de incluir estilo en el código HTML y se hará una pequeña introducción a hojas de estilo CSS (*Cascading Style Sheets*).

5.1. Comenzando con estilos

La forma más habitual de agregarle algo de estilo a los documentos HTML es por medio de atributos para los elementos a los cuales se aplica el estilo. Esta forma solamente se recomienda cuando se están haciendo páginas para navegadores que no tienen soporte para hojas de estilo CSS. Esto ocurre con versiones muy antiguas de navegadores Web como Internet Explorer y Netscape. En la actualidad, los navegadores en sus versiones recientes ya tienen soporte para hojas de estilo, de forma que no hay necesidad de incluir atributos para darle formato al documento, como los de este ejemplo:

Ejemplo 5.1. Agregando atributos de estilo al elemento BODY de un documento

```
<body bgcolor="white" background="textura.jpg" text="black">
```

Los atributos `bgcolor` (color de fondo), `background` (fondo) y `text` (color del texto) pertenecen a atributos de estilo para la etiqueta BODY.

Para comenzar con hojas de estilos, es importante mencionar que en HTML existe un elemento dedicado para mantener la descripción del estilo dado al documento. Este elemento se invoca con la etiqueta `<style>` y se ubica en el encabezado del documento, es decir, dentro del elemento `<head>`. Por ejemplo:

Ejemplo 5.2. Configuración simple de color de letra y de fondo del documento

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Mi título</title>
<style type="text/css">
  body { color: black; background: white; }
</style>
</head>
<body>
Bla bla bla
</body>
</html>
```

En el contenido del elemento `<style>` se ingresan **reglas de estilos**, que tienen una sintaxis especial. Para cada regla, primero va el nombre de la etiqueta a la cual se aplica, luego entre llaves({}) las distintas propiedades del estilo. Cada propiedad se ingresa como clave-valor, donde la clave (el nombre de la propiedad) va al lado izquierdo, el símbolo de dos puntos en el medio, y el valor de la propiedad a la derecha, finalizando con el símbolo de punto y coma.

En el ejemplo anterior, la etiqueta a la que aplica la regla de estilo es `<body>`, por lo que se pone su nombre, y hay dos propiedades, cada una con su respectivo valor. Particularmente son colores, entonces más adelante se explicará cómo asignar valores de colores a las propiedades que los usan.

5.2. Reutilizando estilos

Para reutilizar los mismos estilos en varias páginas (por ejemplo al elaborar un sitio Web), es necesario separar la definición de estilos en un archivo aparte únicamente con la definición de reglas de estilos aplicables para varias páginas, y luego relacionar este archivo con el documento HTML de la siguiente manera:

Ejemplo 5.3. Línea de inserción de estilos definidos en un archivo independiente

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

Los archivos para hojas de estilo se almacenan con la extensión CSS, y al igual que la etiqueta `<style>`, la etiqueta del ejemplo anterior se ubica en el encabezado del documento. El atributo `rel="stylesheet"` le indica al navegador Web que el vínculo corresponde a una hoja de estilos en la ruta marcada por el atributo `href`. En este caso, el documento HTML se encuentra en la misma carpeta del documento CSS.

Se puede enlazar a una hoja de estilos CSS y también tener definidos estilos propios para el documento HTML al mismo tiempo. Si se cruzan las definiciones de reglas, y se tiene la etiqueta `<style>` después de la linea de enlace de la hoja de estilos, tienen prioridad las reglas definidas al interior de `<style>`.

5.3. Configurando estilos comunes

Veamos cómo configurar algunos estilos comunes en hojas de estilo CSS. Para términos de cada ejemplo, se asume que las líneas de definición de estilos van dentro de una etiqueta `<style>` o en un archivo de hoja de estilos.

5.3.1. Márgenes

Las márgenes de un documento se configuran con las propiedades `margin-left` y `margin-right` para la etiqueta `body`. En el ejemplo siguiente se configuran estas márgenes para que sean el 10% del ancho de la ventana en cada lado:

Ejemplo 5.4. Configuración simple de márgenes del documento con CSS

```
body {
  margin-left: 10%;
  margin-right: 10%;
}
```

5.3.2. Sangría

Para hacer sobresalir los encabezados de las secciones del documento del margen del mismo, teniendo en cuenta el ejemplo anterior, se utiliza la propiedad `margin-left` de la siguiente manera:

Ejemplo 5.5. Aplicando sangría a la izquierda para encabezados del documento

```
body {
  margin-left: 10%;
  margin-right: 10%;
}

h1 {
  margin-left: -8%;
}

h2,h3,h4,h5,h6 {
  margin-left: -4%;
}
```

El valor negativo para las márgenes de los encabezados indica en este caso escribir tanto por ciento a la izquierda de la margen del contenedor `<body>`. El encabezado 1 entonces comenzaría a mostrarse al 2% del ancho de la ventana, mientras que los demás encabezados comenzarían al 6% del ancho de la ventana.

5.3.3. Espacios en blanco superiores e inferiores

Supongamos que necesitamos un espacio en blanco arriba y abajo de un encabezado en el documento. El control sobre este espacio se logra con las propiedades `margin-top` y `margin-bottom`. Veamos un ejemplo:

Ejemplo 5.6. Configurando espacios en blanco arriba y abajo de un encabezado de segundo nivel

```
h2 {
  margin-top: 8em;
  margin-bottom: 3em;
}
```

La unidad em utilizada en el ejemplo anterior equivale al tamaño de la fuente, y se puede utilizar de la misma forma que otras unidades como puntos y píxeles.

Para cambiar este estilo a un tipo de encabezado en particular (por ejemplo, encabezado 2 para subsecciones) se utiliza una **clase**, que en el código HTML se describe con el atributo `class` para un elemento cualquiera, y en el código CSS se escribe luego de la etiqueta de la regla, precedida por un punto, así:

Ejemplo 5.7. Configurando espacios en blanco arriba y abajo de un encabezado perteneciente a una clase

En el código CSS:

```
h2.subseccion {
  margin-top: 8em;
  margin-bottom: 3em;
}
```

En el código HTML:

```
<h2 class="subseccion">Mi subsección</h2>
```

5.3.4. Fuentes de texto

Un texto se puede asociar a una fuente de texto, o más específicamente a una familia de fuentes de texto. Teniendo en cuenta que una fuente puede no estar habilitada para un visitante (ya sea por restricción del navegador Web o el sistema operativo como tal), se recomienda especificar una serie de familias de fuentes en orden de preferencia, y especialmente se recomienda finalizar cada serie con una familia genérica (serif, sans-serif, cursive, fantasy, monospace)[1], para que en último caso se utilice una fuente más o menos similar a la preferida.

Ejemplo 5.8. Asociación de familia de fuente para el texto del cuerpo y de los encabezamientos

```
body {
  font-family: Verdana, sans-serif;
}

h1,h2 {
  font-family: Garamond, "Times New Roman", serif;
}
```

El texto del cuerpo del documento en general utilizará la familia Verdana. Si no se encuentra disponible, se utilizará la fuente sans-serif predeterminada del navegador Web. Asimismo, los encabezados de nivel 1 y 2 utilizarán la fuente Garamond, que de no estar disponible, se cambiará por Times New Roman, y si tampoco está disponible, se utilizará una fuente de la familia serif.

Además de configurar la familia de fuentes para un texto, se puede modificar la forma en que el navegador Web presenta textos con formato con cursiva (`<em>`) o negrilla (`<strong>`), como ya se explicó en un capítulo anterior. Veamos el siguiente ejemplo:

[1]Se puede investigar más sobre estas familias de fuentes genéricas para estilos en esta dirección: http://www.w3.org/TR/CSS2/fonts.html#generic-font-families.

Ejemplo 5.9. Cambio en la visualización de texto en negrilla o cursiva aplicando estilos

```
em {
  font-style: italic;
  font-weight: bold;
}

strong {
  text-transform: uppercase;
  font-weight: bold;
}
```

Para el texto con énfasis se aplica además de cursiva, una combinación con negrilla, mientras que para el texto "fuerte" o con negrilla, se aplica una transformación del texto para que aparezca siempre en mayúscula sostenida.

También se puede configurar el tamaño de la fuente para diferentes etiquetas, en proporción al tamaño de la fuente establecida para el cuerpo del documento. Por ejemplo, para redefinir el tamaño del texto de los encabezados de un documento HTML:

Ejemplo 5.10. Redefinición del tamaño de la fuente para los encabezados

```
h1 {
  font-size: 200%;
}

h2 {
  font-size: 150%;
}

h3 {
  font-size: 100%;
}
```

Se recomienda utilizar porcentajes en vez de puntos para estas reglas, ya que puede variar para visitantes que configuren fuentes más o menos grandes de acuerdo a su visión.

5.3.5. Bordes y fondos

Se pueden crear bordes alrededor de elementos como encabezados, listas, párrafos y otros, encerrando estas etiquetas en una etiqueta `<div>` y creando reglas de estilo para esta última, así:

Ejemplo 5.11. Estilo de borde simple utilizando el elemento DIV y hojas de estilo

En el código CSS:

```
div.caja {
  border: solid;
  border-width: thin;
  width: 100%;
}
```

En el código HTML:

```
<div class="caja">
...
</div>
```

Los elementos DIV marcados con la clase "caja" tendrán borde y ocuparán el 100% del ancho del documento.

Así como se agregó un borde a un fragmento del documento, se le puede asociar un color de fondo para resaltarlo dentro del contenido del texto, también encerrando el fragmento del documento dentro de un elemento DIV. Por ejemplo:

Ejemplo 5.12. Color de fondo para un fragmento de código HTML encerrado en un elemento DIV

```
div.coloreado {
  background: rgb(204,204,255);
  padding: 0.5em;
  border: none;
}
```

5.3.6. Colores

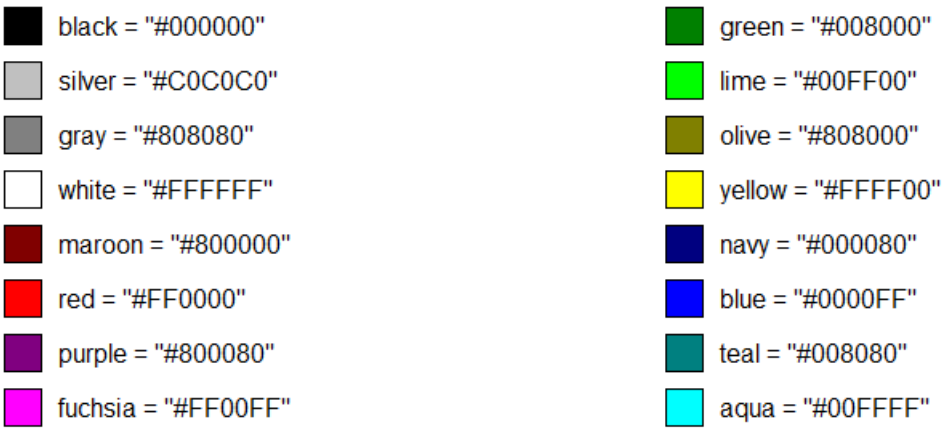
Representación de los colores

Hasta ahora se han utilizado algunas propiedades relacionadas con colores, y se ha visto que se pueden utilizar por su nombre en inglés (si lo tienen), o especificando el color con la combinación de valores de la escala RGB (*Red Green Blue*). La primera de estas dos formas solamente sirve para 16 colores que tienen nombre (ver Figura 5.1, “Colores con nombre para hojas de estilos, y sus valores hexadecimales”, original de Dave Ragget), y a pesar de ésto no se recomienda su uso por razones de portabilidad entre navegadores; la segunda forma necesita una combinación de 3 valores que varían entre 0 y 255 para cada color de la escala (rojo, verde y azul), por ejemplo: `rgb(204,204,255)`.

Además de estas dos formas, existe una tercera forma que consiste en escribir el código hexadecimal para el color, por ejemplo, `#000000` para el negro. Este valor consta de 3 partes, también correspondientes a la escala RGB. Los 2 primeros números son el valor hexadecimal para la escala de rojo, luego 2 para la escala de verde y los últimos 2 para la escala de azul.

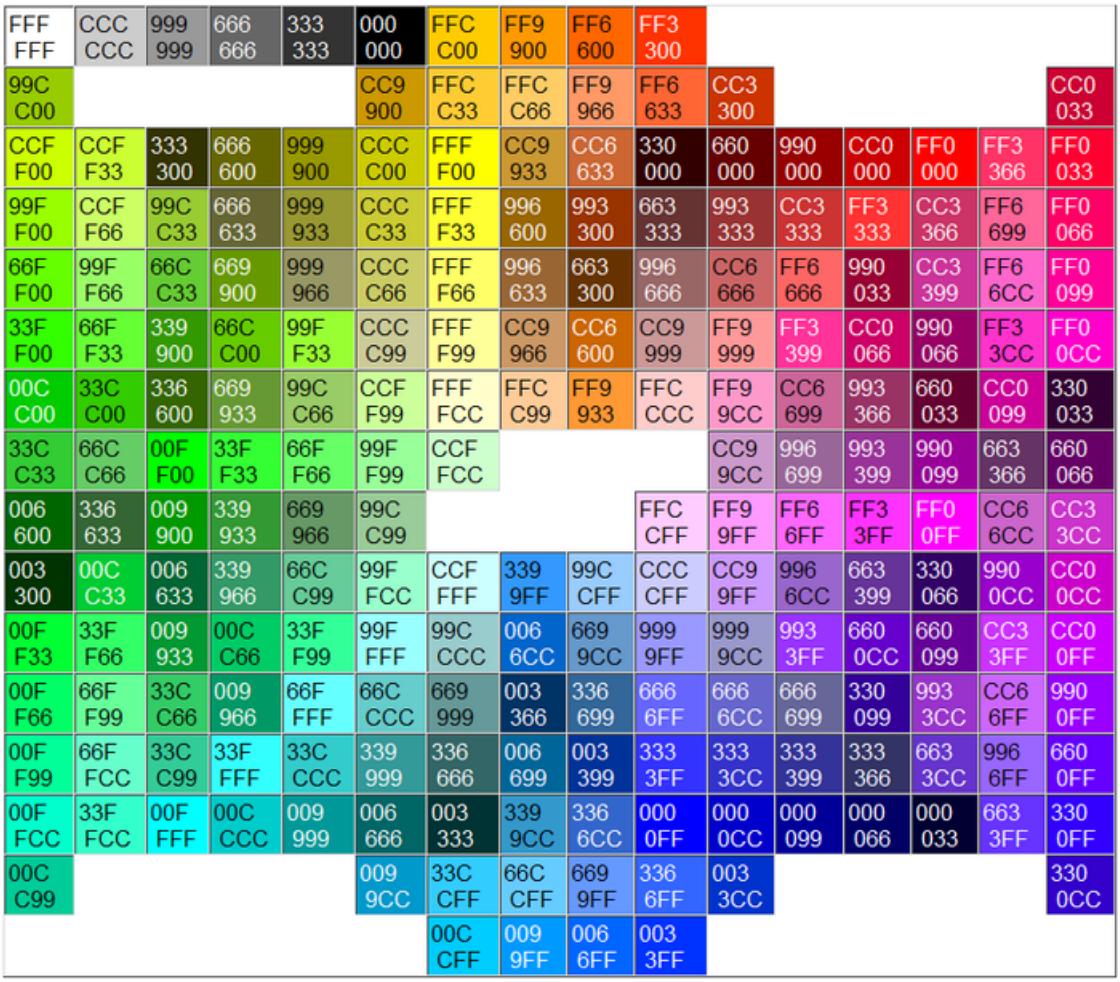
Los valores hexadecimales de los colores se pueden obtener utilizando una calculadora y transformando el valor decimal entre 0 y 255 al valor hexadecimal.

Figura 5.1. Colores con nombre para hojas de estilos, y sus valores hexadecimales



Los computadores actuales tienen pantallas que soportan incluso millones de colores, de forma que se puede jugar con las combinaciones posibles de las escalas de rojo, verde y azul. Sin embargo, todavía hay pantallas para menor cantidad de colores, como los dispositivos móviles, e incluso computadores obsoletos funcionando con 256 colores. Si quisiéramos incluir entre los visitantes aquellos con esta clase de pantallas, hay que utilizar colores que puedan ver tal cual como se definen, no colores parecidos. En la Figura 5.2, “Colores seguros para cualquier navegador Web” se muestra una matriz de colores considerados "seguros" entre navegadores y dispositivos.

Figura 5.2. Colores seguros para cualquier navegador Web



Colores de enlaces

Se puede cambiar el formato de los enlaces de un documento por medio de estilos. Hay que tener en cuenta que los enlaces funcionan como visitado/no-visitado, y además manejan eventos tales como los utilizados para hacer *scripting* en el capítulo anterior (cuando se hace clic, cuando el cursor está sobre el enlace, etc.). Estos estados se pueden representar en CSS como se muestra a continuación:

Ejemplo 5.13. Cambiando el color de los enlaces en un documento

```
/* Enlaces sin visitar */
:link {
  color: rgb(0, 0, 153);
}

/* Enlaces visitados */
:visited {
  color: rgb(153, 0, 153);
}

/* Cuando se hace clic en el enlace */
a:active {
  color: rgb(255, 0, 102);
}

/* Cuando el cursor está sobre el enlace */
a:hover {
  color: rgb(0, 96, 255);
}
```

Así como se pueden cambiar los colores de los enlaces, también se les puede quitar el subrayado. Sin embargo, el común de la gente espera que lo que aparece como subrayado y en azul en un documento es un enlace, por lo que cambiar este estilo puede hacer perder a los visitantes. Esta práctica solo se recomienda cuando es necesaria una transformación para la estética del documento.

Ejemplo 5.14. Eliminando el subrayado de los enlaces

En el código CSS:

```
a.plano {
  text-decoration: none;
}
```

En el código HTML:

```
Esto es un <a class="plano" href="#">enlace sin subrayar</a>
```

Capítulo 6. Marcos

Los marcos son formas de organización de documentos HTML a modo de vistas múltiples. La idea de los marcos es mantener cierta información visible mientras se navega en otra parte del documento o en páginas relacionadas. Esta práctica se realizaba mucho antiguamente para mantener visibles los menús de navegación, mientras en un marco central se cargan las distintas páginas de un sitio Web.

6.1. Comenzando el trabajo con marcos

Para trabajar con marcos en los documentos HTML se hace necesaria una redefinición del tipo de documento para que se permita el uso de estos marcos. A continuación se presenta un documento de ejemplo que hace uso de marcos:

Ejemplo 6.1. Documento HTML definido para contener 3 marcos

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
    "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<title>Documento con marcos</title>
</head>
<frameset cols="20%, 80%">
  <frameset rows="100, 200">
      <frame src="marco_superior_izquierdo.html">
      <frame src="marco_inferior_izquierdo.html">
  </frameset>
  <frame src="marco_central.html">
  <noframes>
      <p>Este documento contiene:</p>
      <ul>
         <li><a href="marco_superior_izquierdo.html">Un logo</a>
         <li><a href="marco_inferior_izquierdo.html">Un men&uacute;</a>
         <li><a href="marco_central.html">Un marco central</a>
      </ul>
  </noframes>
</frameset>
</html>
```

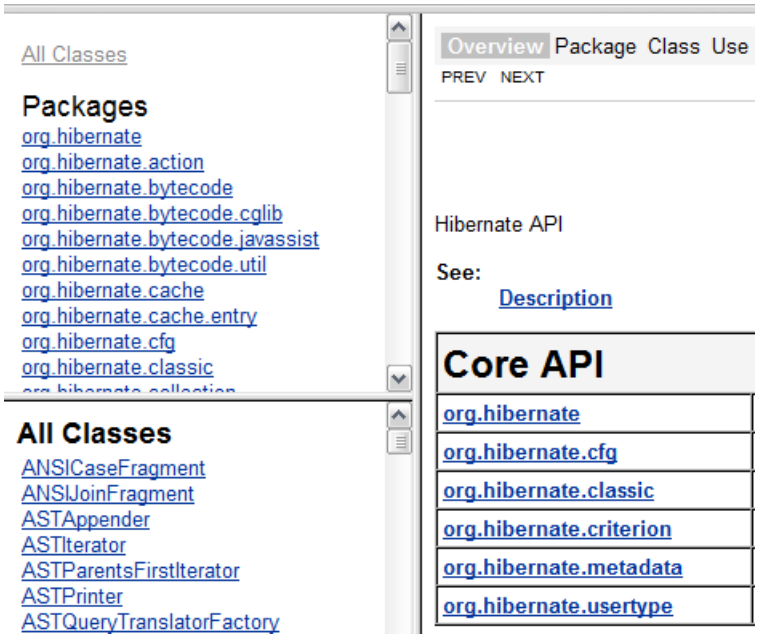
Se definen 3 marcos, primero separando la ventana en dos columnas (primera ocupa 20% y segunda ocupa 80%). Luego, la primera columna se divide en dos marcos, mientras que en la segunda columna va el marco central.

Como se observa, el documento no lleva elemento BODY, sino un elemento FRAMESET en sustitución, y cada FRAMESET puede dividirse en uno o más del mismo tipo, o contener marcos simples, es decir, elementos FRAME. La organización más habitual de las páginas con marcos es ésta con 3 divisiones de la ventana. El contenido del elemento NOFRAMES se muestra cuando el navegador no tiene soporte para marcos. En la actualidad, los navegadores más populares todavía soportan marcos, a pesar de que su uso ya no es tan común como antes.

Para que el documento pueda contener marcos, debe cambiarse la definición del tipo de documento a "HTML 4.01 Frameset" y ajustar la dirección de la referencia DTD a la referencia adecuada para la especificación con marcos.

Todo el contenido que se inserta dentro de la etiqueta <body> puede colocarse dentro de las etiquetas `<frameset>` y `<frame>`, aunque para hacer el documento más organizado, el contenido de cada marco se especifica en un documento aparte, y se referencia por medio del atributo `src` del elemento FRAME.

Figura 6.1. Documento HTML en línea que hace uso de marcos



6.2. Diseño de marcos

En la sección anterior se explicaba el cambio que se debe hacer en el documento HTML para poder definir marcos en su interior. Asimismo, se explicaba que se tiene un elemento FRAMESET (conjunto de marcos) como base, y que puede contener otros elementos FRAMESET para agrupar aún más algunos elementos FRAME, o se pueden tener estos elementos FRAME directamente bajo el conjunto principal como se mostraba en el ejemplo.

Es importante siempre especificar la forma en que se distribuye un conjunto de marcos a nivel de filas y columnas. La cantidad y la distribución de las filas de un conjunto de marcos se define mediante el atributo `rows` del FRAMESET, mientras que las columnas se especifican con el atributo `cols`. El valor de estos atributos es la distribución o espacio que va a tomar con respecto al espacio total de la ventana donde se muestra el documento, por cada fila y columna, separando por coma cada una de ellas. La distribución de espacio de cada fila o columna en un FRAMESET se define en términos de porcentajes, píxeles o multiplicadores.

A continuación se describen algunos atributos configurables de los marcos en HTML:

- `scrolling`. Este atributo puede tomar valores de *yes* o *no*, e indica si se permite o no el uso de barra de desplazamiento para el marco.
- `noresize`. La presencia de este atributo (no requiere valor), indica que el marco no puede ser redimensionado.
- `frameborder`. Un valor de 1 para este atributo indica que el navegador debe dibujarle un borde al marco, mientras que 0 indica que no se debe dibujar un borde.
- `name`. Identificador del marco con respecto al documento.

Veamos un ejemplo algo más complicado:

Ejemplo 6.2. Documento con marcos y atributos adicionales para cada marco

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<title>Documento con marcos</title>
</head>
<frameset cols="33%,33%,33%">
  <frameset rows="*,200">
      <frame src="contenido_marco_1.html" scrolling="no">
      <frame src="contenido_marco_2.html"
                marginwidth="10" marginheight="15"
                noresize>
  </frameset>
  <frame src="contenido_marco_3.html" frameborder="0">
  <frame src="contenido_marco_4.html" frameborder="0">
</frameset>
</html>
```

Se dibujan 4 marcos en 3 columnas casi de igual proporción. La primera columna se divide en dos marcos, de donde el primero no permite desplazamiento y el segundo no permite redimensionado y tiene un alto fijo de 200 píxeles.

6.3. Especificación de marcos objetivo

Para especificar en qué marco se va a cargar un enlace luego de que se hace clic sobre uno, se utiliza el atributo `name` del elemento FRAME. El valor de este atributo me permite encontrar un marco dentro de un documento, de forma que el documento al que apunta un enlace pueda cargarse en el marco adecuado. Para hacer esta referencia, en el enlace se debe agregar el atributo `target` para indicar el marco objetivo para cargar el nuevo documento. Veamos un ejemplo:

Ejemplo 6.3. Utilización de enlaces con marcos objetivo para la carga del documento

En el documento `index.html` se tiene este fragmento de código:

```
<frameset cols="20%,80%">
  <frameset rows="30%,70%">
    <frame src="overview-frame.html" name="packageListFrame" title="All Packages">
    <frame src="allclasses-frame.html" name="packageFrame"
           title="All classes and interfaces (except non-static nested types)">
  </frameset>
  <frame src="overview-summary.html" name="classFrame"
         title="Package, class and interface descriptions">
</frameset>
```

En el documento `overview-frame.html` se tiene un enlace para cargar un documento en una ruta dada, en el marco `packageFrame`:

```
<a href="org/hibernate/package-frame.html" target="packageFrame">org.hibernate</a>
```

Si no se especifica un marco objetivo en un enlace, el documento cargará en el mismo marco donde se encuentra el enlace.

6.4. Marcos entre líneas

Una forma alternativa de utilizar marcos en un documento HTML, que no requiera de dividir la ventana sino simplemente dibujar un documento HTML en medio de otro es por medio de marcos entre líneas, que se definen como elementos IFRAME, es decir, con la etiqueta `<iframe>`. Estos marcos entre líneas funcionan de forma análoga a los elementos OBJECT, que se utilizaban para insertar videos de servicios públicos como YouTube o Vimeo; también mantienen características de los marcos comunes, como por ejemplo, el atributo `src` para indicar la fuente del marco, el atributo `scrolling` y el atributo `frameborder`, entre otros. Un atributo que no sea usa es el atributo `noresize`, ya que los marcos entre líneas no se redimensionan, de forma que sobra definirlo.

Ejemplo 6.4. Marco entre líneas simple (IFRAME)

```
<iframe src="contenido.html" width="400" height="500"
             scrolling="auto" frameborder="1">
Este texto se muestra si el navegador Web no tiene soporte de marcos.
</iframe>
```

Capítulo 7. Formularios

Los formularios en HTML son partes de documentos que además del contenido normal visto hasta ahora, pueden contener elementos especiales llamados **controles** y otros llamados **etiquetas**. Estos controles pueden ser campos de texto, listas desplegables, botones, cajas de verificación, entre otros, y tienen la característica de que sirven como medio de entrada para información ingresada por el usuario.

En este capítulo se explicará lo relacionado al manejo de estos formularios en HTML.

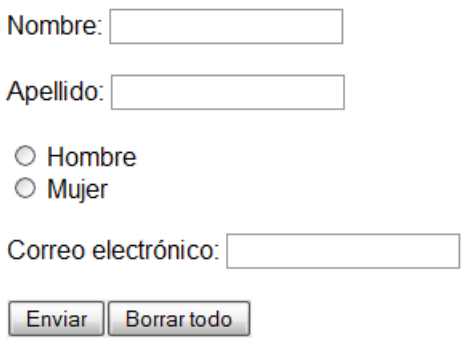
7.1. Trabajando con formularios

Los formularios se especifican en HTML por medio de la etiqueta <form>. Esta etiqueta funciona como contenedor de otros elementos, que pueden ser el contenido que hemos visto hasta ahora: títulos, párrafos, listas, tablas, etc.; también puede contener elementos propios de formularios, como controles y etiquetas, como se observa en el siguiente ejemplo:

Ejemplo 7.1. Formulario simple con campos de texto, botones de selección y botones de confirmación y limpieza

```
<body>
 ...
 <form action="http://ejemplo.com/registrar_usuario" method="post">
    <p>
    <label for="nombre">Nombre: </label>
    <input type="text" id="nombre"><br>
    <label for="apellido">Apellido: </label>
    <input type="text" id="apellido"><br>
    <input type="radio" name="genero" value="m"> Hombre<br>
    <input type="radio" name="genero" value="f"> Mujer<br>
    <label for="email">Correo electrónico: </label>
    <input type="text" id="email"><br>
    <input type="submit" value="Enviar"><input type="reset" value="Borrar todo">
    </p>
 </form>
 ...
</body>
```

Figura 7.1. Formulario simple con campos de texto, botones de selección y botones de confirmación y limpieza



Todo formulario necesita de los atributos `action` y `method`. El atributo `action` contiene la dirección URI (*Universal Resource Identifier*) del punto a donde se envía la información capturada en el formulario para su procesamiento, es decir, un servicio Web, un servicio de envío de correo, etc. Por otro lado, el atributo `method` indica la forma en que se envía la información del formulario. Existen dos formas de enviar la información del formulario:

- Método GET: La información se envía como parte de la dirección URI del atributo action, agregando un signo ? al final de ésta y cada control con su nombre y valor respectivo separado con un &.

- Método POST: La información del formulario se envía como parte del cuerpo de la solicitud. Cuando se utiliza este método, es recomendable también utilizar el atributo `enctype` del elemento FORM para indicar el tipo de contenido que se está enviando. Éste método se utiliza para enviar datos que contienen archivos o imágenes.

Los datos del formulario se envían con el **nombre del control** de donde proviene cada dato, es decir, para identificar un dato, es necesario que cada control tenga un valor para el atributo `name`. Si el control no tiene nombre asignado, el servidor como tal se encarga de asignarle uno.

7.2. Controles

Los controles son los elementos en HTML que permiten ingresar y capturar entradas por parte del usuario. A continuación se explica cómo utilizar las diferentes clases de controles más comunes en los formularios de HTML.

Muchos de los controles más comunes en HTML se crean utilizando el elemento INPUT en HTML, es decir, utilizando la etiqueta `<input>`. Estos controles son:

- Campos de texto
- Campos de contraseñas
- Cajas de verificación
- Botones de selección
- Botones de confirmación
- Botones de confirmación con imagen
- Botones de limpieza del formulario
- Botones simples
- Selección de archivo

Otros controles se crean utilizando otras etiquetas, como los menús y las áreas de texto. La Figura 7.1, “Formulario simple con campos de texto, botones de selección y botones de confirmación y limpieza” es un ejemplo de formulario utilizando campos de texto (nombre, apellido y correo electrónico), botones de selección (género), botones de confirmación (enviar) y botones de limpieza del formulario (borrar todo).

En todos los casos en que se ingresa texto, se puede especificar el valor inicial del campo utilizando el atributo `value` del elemento.

7.2.1. Campos de texto

Para crear un campo de texto en un formulario se ingresa dentro de un formulario un código como éste:

```
<input type="text" name="nombre">
```

7.2.2. Áreas de texto

Funcionan igual que los campos de texto, solo que en este caso se permite ingresar varias líneas de texto. Se puede configurar el ancho y alto del texto por número de caracteres (ancho) y número de líneas (alto). También se puede configurar un valor inicial. Veamos un ejemplo:

```
<textarea name="descripcion" rows="20" cols="80">
[Mi descripción]
```

```
</textarea>
```

7.2.3. Campos de contraseñas

La diferencia entre un campo de texto y un campo de contraseñas, es que el texto se muestra oculto con caracteres especiales como el asterisco (*) o puntos de relleno. La forma de declararlo es ésta:

```
<input type="password" name="contrasena">
```

7.2.4. Cajas de verificación

Las cajas de verificación o *check boxes* en inglés son controles de entrada simples, que solo se activan o desactivan para indicar si o no a un criterio o pregunta. Estas cajas de verificación se escriben de la siguiente manera:

```
<input type="checkbox" name="enviar_mensajes">
```

7.2.5. Botones de selección

Los botones de selección son listas de opciones de las cuales se debe seleccionar solo una de ellas, y cada opción aparece con una descripción y al lado un botón de selección generalmente de forma circular. Se observa claramente un ejemplo de estos botones en la Figura 7.1, “Formulario simple con campos de texto, botones de selección y botones de confirmación y limpieza”. Para crear botones de selección se ingresa un código como éste. El valor se asigna a cada opción de forma única (no se pueden repetir los valores) utilizando el atributo `value`.

```
<input type="radio" name="genero" value="m"> Hombre <br>
<input type="radio" name="genero" value="f"> Mujer <br>
```

7.2.6. Botones de confirmación

Usualmente los formularios tienen un botón al final para enviar la información ingresada. A este botón se le conoce como botón de confirmación. Una vez se activa este botón, se envía la información a la dirección indicada en el atributo `action` del elemento FORM que contiene el botón. Los botones de confirmación se escriben de esta forma:

```
<input type="submit" value="Enviar">
```

El texto que va dentro del botón se especifica con el atributo `value`.

7.2.7. Botones de confirmación con imagen

Funcionan igual que los botones de confirmación simples, solo que en la declaración se especifica una ruta para la imagen que se muestra, además de un texto alternativo para el caso en que no se pueda mostrar la imagen. Un ejemplo:

```
<input type="image" src="boton_enviar.png" alt="Enviar">
```

7.2.8. Botones de limpieza de formulario

Los botones de limpieza de formulario se encargan de borrar los datos ingresados al formulario y dejarlo con los valores iniciales especificados para cada campo. Se declaran en el código HTML de la siguiente manera:

```
<input type="reset" value="Borrar todo">
```

7.2.9. Selección de archivo

Uno de los controles en HTML es un control para la selección de un archivo para ser referido en un formulario. Este control tiene la apariencia tradicional de un botón "Examinar" (o similares) con un campo de texto donde se diligencia la ubicación de un archivo, ya sea manualmente o por medio del diálogo que aparece con el botón. El campo se crea de esta manera:

```
<input type="file" id="foto">
```

7.2.10. Menús o listas seleccionables

El elemento SELECT permite crear menús, también conocidos como listas seleccionables. Cada menú consta de ítems que se ingresan por medio de los elementos OPTION. También se puede hacer que por defecto aparezca una o varias de estas opciones como seleccionadas al momento de cargar el formulario o de limpiarlo. Al final se pueden escoger una o varias opciones, dependiendo de si se crea o no el elemento SELECT utilizando el atributo `multiple`. Veamos un ejemplo:

```
<select multiple size="6" name="finalistas">
  <option selected value="0"> -Seleccione sus finalistas- </option>
  <option value="juan">Juan G&oacute;mez</option>
  <option value="pedro">Pedro P&eacute;rez</option>
  <option value="marta">Marta S&aacute;nchez</option>
  <option value="maria">Mar&iacute;a Gonz&aacute;lez</option>
  <option value="pablo">Pablo Ram&iacute;rez</option>
  <option value="ana">Ana Ruiz</option>
</select>
```

En el ejemplo, se muestra una lista de selección múltiple, que se muestra muy diferente a una lista de selección simple (una sola línea). Para hacerla simple, se elimina de la declaración el atributo `multiple`. El atributo size indica la cantidad de ítems que se muestran inicialmente en la pantalla.

También se pueden agrupar los ítems dentro de la lista para facilitar la selección. Para agruparlos, se utiliza el elemento OPTGROUP así:

```
<select multiple size="6" name="finalistas">
  <option selected value="0"> -Seleccione sus finalistas- </option>
  <optgroup label="Hombres">
    <option value="juan">Juan G&oacute;mez</option>
    <option value="pedro">Pedro P&eacute;rez</option>
    <option value="pablo">Pablo Ram&iacute;rez</option>
  </optgroup>
  <optgroup label="Mujeres">
    <option value="marta">Marta S&aacute;nchez</option>
    <option value="maria">Mar&iacute;a Gonz&aacute;lez</option>
    <option value="ana">Ana Ruiz</option>
  </optgroup>
</select>
```

7.2.11. Campos ocultos

Se pueden incluir algunos datos en el formulario sin necesidad de mostrarlos en el formulario. Estos datos son útiles por ejemplo para pasar identificadores del usuario que llena el formulario y otros datos similares que no se muestran, pero que deben ser enviados con el formulario para facilitar el procesamiento o para evitar perder información si el formulario pertenece a uno de varios pasos que incluyen otros formularios.

Por ser datos ocultos, la información ya debe estar disponible de antemano o al momento de cargar el formulario. Por esta razón, es necesario incluir el valor del campo, como se muestra en el siguiente ejemplo:

```
<input type="hidden" name="codigo_usuario" value="80860">
```

7.3. Etiquetas

Algunos controles incluyen un espacio de texto que sirve como etiqueta para visualmente indicar a qué corresponde dicho control, por ejemplo los botones seleccionables en la Figura 7.1, “Formulario simple con campos de texto, botones de selección y botones de confirmación y limpieza”. Sin embargo, otros controles no incluyen esta etiqueta, de forma que hay que agregársela para indicarle al usuario de qué se trata el control. Para estos casos, se agrega un elemento LABEL antes (usualmente) o después del control, dependiendo de la organización del formulario.

Veamos un ejemplo con un formulario más avanzado:

Ejemplo 7.2. Formulario avanzado con etiquetas y mas tipos de campos

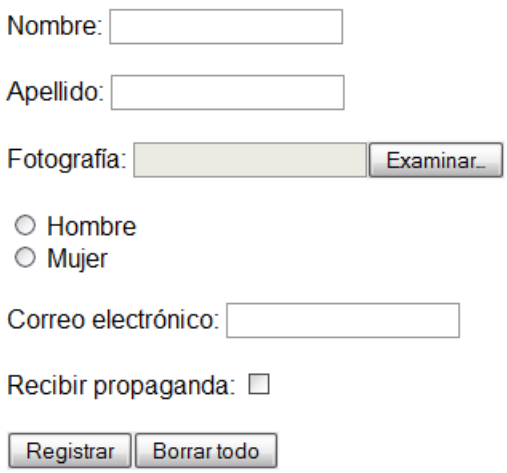
```
<form action="http://ejemplo.com/registrar_usuario" method="post">
   <p>
   <label for="nombre">Nombre: </label>
   <input type="text" id="nombre"><br><br>
   <label for="apellido">Apellido: </label>
   <input type="text" id="apellido"><br><br>
   <label for="foto">Fotografía: </label>
   <input type="file" id="foto"><br><br>
   <input type="radio" name="genero" value="m"> Hombre<br>
   <input type="radio" name="genero" value="f"> Mujer<br><br>
   <label for="email">Correo electrónico: </label>
   <input type="text" id="email"><br><br>
   <label for="propaganda">Recibir propaganda: </label>
   <input type="checkbox" id="propaganda"><br><br>
   <input type="submit" value="Registrar"><input type="reset" value="Borrar todo">
   </p>
</form>
```

Para los campos o controles que no incluyen etiquetas se crea un elemento LABEL antes con un texto descriptivo, y se utiliza el atributo `for` para enlazar la etiqueta con el identificador del campo (`id`).

Para enlazar una etiqueta con un campo o control, se puede omitir el atributo for, pero con la condición que el campo quede definido dentro de la etiqueta, como se muestra enseguida. En ambos casos el resultado visualmente es el mismo, como se muestra en la Figura 7.2, “Formulario avanzado con etiquetas y mas tipos de campos”.

```
<label>Nombre: <input type="text"></label>
```

Figura 7.2. Formulario avanzado con etiquetas y mas tipos de campos



7.4. Agrupaciones

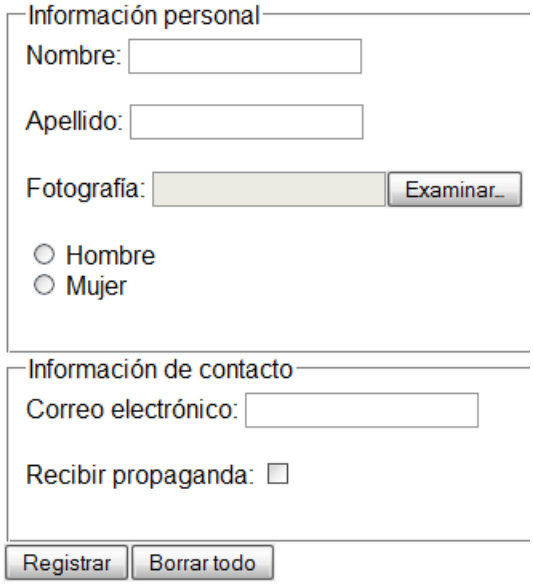
Los controles y etiquetas en un formulario se pueden agrupar, especialmente si el formulario es muy grande, aunque se puede hacer igualmente con formularios pequeños para organizar la información tanto visualmente como semánticamente (significado). Las agrupaciones de estos controles y sus etiquetas se hace incluyendo éstos dentro de un elemento contenedor FIELDSET, que habitualmente tendrá un título, que se coloca utilizando un elemento LEGEND antes de ubicar la primera etiqueta o control dentro de dicho elemento.

Ejemplo 7.3. Formulario avanzado con agrupaciones de etiquetas y controles por tema

```
<form action="http://ejemplo.com/registrar_usuario" method="post">
 <p>
 <fieldset>
   <legend>Información personal</legend>
   <label for="nombre">Nombre: </label>
   <input type="text" id="nombre"><br><br>
   <label for="apellido">Apellido: </label>
   <input type="text" id="apellido"><br><br>
   <label for="foto">Fotografía: </label>
   <input type="file" id="foto"><br><br>
   <input type="radio" name="genero" value="m"> Hombre<br>
   <input type="radio" name="genero" value="f"> Mujer<br><br>
 </fieldset>
 <fieldset>
   <legend>Información de contacto</legend>
   <label for="email">Correo electrónico: </label>
   <input type="text" id="email"><br><br>
   <label for="propaganda">Recibir propaganda: </label>
   <input type="checkbox" id="propaganda"><br><br>
 </fieldset>
 <input type="submit" value="Registrar"><input type="reset" value="Borrar todo">
 </p>
</form>
```

En un navegador Web el resultado sería como se muestra en la Figura 7.3, “Formulario avanzado con agrupaciones de etiquetas y controles por tema”.

Figura 7.3. Formulario avanzado con agrupaciones de etiquetas y controles por tema



Capítulo 8. Scripts

Con *scripting* se describe la actividad de incluir *scripts* en un documento HTML para hacerlo más dinámico e interesante. En este capítulo se estudiará lo necesario para comenzar a trabajar con *scripts* en nuestros documentos para lograr este efecto.

8.1. ¿Qué es un *script*?

En HTML un *script* es un fragmento de código fuente que puede acompañar los documentos o puede estar incluído en él. Este código se ejecuta cuando se carga el documento o cuando se activa algún evento asociado al documento o a alguno de los elementos que lo componen.

Los *scripts* facilitan la extensión de los documentos HTML haciéndolos dinámicos y/o interactivos de alguna de estas formas:

- Modificando el contenido de un documento dinámicamente
- Acompañando a los formularios para validar cada dato que es ingresado. También se pueden ingresar dinámicamente los valores en los campos de un formulario con base en valores de otros campos.
- Se pueden activar *scripts* a medida que ocurren eventos en un documento (cargar, obtener el foco, mover el cursor del ratón, etc.).
- Se pueden vincular los *scripts* con controles de formularios para crear componentes visuales de forma dinámica.

Los scripts se ejecutan siempre en el computador con el navegador Web desde donde se carga el documento HTML, a pesar de que las páginas Web y los *scripts* se encuentren alojados en un servidor Web en Internet.

Se pueden clasificar entonces los scripts en dos grandes grupos:

- *Scripts* que se ejecutan con la carga de un documento HTML.
- *Scripts* que se ejecutan con la activación de eventos asociados con elementos al interior de un documento HTML.

8.2. Creando documentos con scripts

Para crear un *script* dentro de un documento se debe utilizar el elemento SCRIPT tantas veces como sea necesario, dependiendo de su utilización, siempre y cuando este elemento quede dentro de un elemento HEAD o un elemento BODY. Sin embargo, los *scripts* a utilizar en una página Web también se pueden definir en archivos aparte, y en la definición del elemento SCRIPT se especifica la ubicación del archivo con el *script* utilizando el atributo `src`.

En el capítulo 4 veíamos ejemplos de cómo crear los elementos SCRIPT dentro del documento HTML, y veíamos que existen varios lenguajes de programación con los cuales se pueden programar estos *scripts*:

- JavaScript
- Visual Basic Script
- TCL

Es importante que para que estos *scripts* se ejecuten se tenga un navegador Web que permita interpretar el código fuente incluído allí. Hoy en día, el lenguaje de programación más utilizado para escribir estos *scripts* es **JavaScript**, porque la mayoría de navegadores Web ofrece soporte para este lenguaje, mientras que los soportes para los demás lenguajes son escasos o dependientes de un sistema operativo, lo que hace que no esté disponible el efecto que pueda provocar el *script* para cualquier visitante.

Se recomienda siempre especificar el lenguaje en que se encuentra el contenido de cada elemento SCRIPT, a pesar de que se pueda especificar un lenguaje por defecto para todos los scripts.

Ejemplo 8.1. Creación de scripts en lenguajes diferentes y en distintas partes de un documento HTML (Original del W3C)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
     "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Un documento con elementos SCRIPT</title>
  <meta http-equiv="Content-Script-Type" content="text/tcl">
  <script type="text/vbscript" src="http://ejemplo.com/progs/vbcalc" />
</head>
<body>
  <script type="text/javascript">

  ...código JavaScript...

  </script>
</body>
</html>
```

En el elemento META del encabezamiento del documento se especifica que el lenguaje por defecto de los *scripts* del documento es TCL, pero se crean 2 *scripts* en el documento: el primero utiliza Visual Basic Script (en el encabezamiento) y se encuentra en una ubicación aparte, mientras que el segundo se encuentra en JavaScript y se especifica dentro del mismo documento (en el cuerpo).

Al igual que con las hojas de estilos en CSS, se pueden reutilizar los *scripts* creando el código fuente en archivos independientes e invocándolos en el encabezamiento de los documentos HTML que los requieran.

8.3. Activación y ejecución de scripts

Como se explicó en la primera parte de este capítulo, los scripts se pueden activar al momento de cargar un documento (concretamente el cuerpo) o al momento que se activa un evento sobre alguno de los elementos que compone el cuerpo de un documento HTML. En ambos casos, se debe especificar ya sea para el cuerpo del documento o para un elemento, un atributo especial de acuerdo al tipo de evento, y el valor de este atributo debe ser una línea de código con la invocación a alguna parte del script, bien puede ser una función.

A continuación se encuentra el listado de cada uno de los posibles eventos existentes en HTML:

- `onload`. Ocurre al terminar de cargar una ventana o un conjunto de marcos (elemento FRAMESET). Solo se puede utilizar este atributo en elementos BODY y FRAMESET.
- `onunload`. Ocurre cuando un navegador quita o cambia un documento de la ventana. Solo se puede utilizar este atributo en elementos BODY y FRAMESET.
- `onclick`. Ocurre cuando se hace clic sobre un elemento. Se puede utilizar con la mayoría de elementos.
- `ondblclick`. Ocurre cuando se hace doble clic sobre un elemento. Se puede utilizar con la mayoría de elementos.
- `onmousedown`. Ocurre cuando el puntero se encuentra sobre un elemento y se presiona un botón del ratón. Se puede utilizar con la mayoría de elementos.
- `onmouseup`. Ocurre cuando se está liberando un botón del ratón sobre algún elemento. Se puede utilizar con la mayoría de elementos.
- `onmouseover`. Ocurre cuando se pasa el puntero del ratón sobre un elemento. Se puede utilizar con la mayoría de elementos.

- `onmousemove`. Ocurre cuando el puntero del ratón se mueve mientras éste está sobre un elemento. Se puede utilizar con la mayoría de elementos.
- `onmouseout`. Ocurre cuando el puntero del ratón sale de un elemento. Se puede utilizar con la mayoría de elementos.
- `onfocus`. Ocurre cuando un elemento del documento obtiene el foco, ya sea por medio del puntero o por medio de la navegación por tabulador. Se puede utilizar con los elementos A, AREA, LABEL, INPUT, SELECT, TEXTAREA y BUTTON.
- `onblur`. Ocurre cuando un elemento del documento pierde el foco, ya sea por medio del puntero o por medio de la navegación por tabulador. Se puede utilizar con los elementos A, AREA, LABEL, INPUT, SELECT, TEXTAREA y BUTTON.
- `onkeypress`. Ocurre cuando se presiona y libera una tecla estando sobre un elemento. Se puede utilizar con la mayoría de elementos.
- `onkeydown`. Ocurre cuando se presiona una tecla estando sobre un elemento. Se puede utilizar con la mayoría de elementos.
- `onkeyup`. Ocurre cuando se libera una tecla que estaba presionada estando sobre un elemento. Se puede utilizar con la mayoría de elementos.
- `onsubmit`. Ocurre cuando se envían los datos de un formulario haciendo clic en el botón de confirmación. Solo se puede utilizar con elementos FORM.
- `onreset`. Ocurre cuando se limpian los datos de un formulario haciendo clic en el botón de limpieza. Solo se puede utilizar con elementos FORM.
- `onselect`. Ocurre cuando se selecciona un texto en un área de texto o campo de texto. Solo se puede utilizar en elementos INPUT y TEXTAREA.
- `onchange`. Ocurre cuando un control de un formulario pierde el foco y su valor cambió mientras tenía el foco. Solo se puede utilizar en elementos INPUT, TEXTAREA y SELECT.

Veamos un ejemplo:

Ejemplo 8.2. Activación de un evento para un script

```
<input name="nombre_usuario" onblur="validarNombreUsuario(this.value)">
```

Al momento de que el campo pierde el foco, se invoca la función `validarNombreUsuario(valor)`,que debe estar implementada en algún elemento SCRIPT del documento HTML.

Otro ejemplo:

Ejemplo 8.3. Modificación dinámica de un documento utilizando JavaScript

```
<head>
  <title>Un t&iacute;tulo</title>
  <script type="text/javascript" src="js/scripts.js"></script>
</head>
<body>
...
<script type="text/javascript">
  document.write("<p>Hola Mundo<\/p>")
</script>
...
</body>
```

El *script* dentro del cuerpo escribe un párrafo al momento de cargar el documento utilizando código HTML.

8.4. Documentos para navegadores que no soportan scripts

En algunos casos se hace necesario considerar que los visitantes de nuestras páginas diseñadas en HTML utilizan navegadores Web especiales o sin soporte para *scripts*. En esa eventualidad, todo el código que se ejecuta o se procesa dentro de los elementos SCRIPT del documento no va a tener efecto, por lo que se hace necesario incluir en el documento HTML elementos NOSCRIPT, donde se ingrese todo el código HTML que pueda faltar para que el resultado sea el mismo, pero posiblemente de forma estática.

En el ejemplo de la sección anterior donde se modificaba de forma dinámica el documento, si no se tiene soporte para *scripts*, habría que tener en el código algo como ésto:

Ejemplo 8.4. Inclusión de fragmentos de código HTML para prevención de no-soporte para scripts

```
<head>
  <title>Un t&iacute;tulo</title>
  <script type="text/javascript" src="js/scripts.js"></script>
</head>
<body>
...
<script type="text/javascript">
  document.write("<p>Hola Mundo<\/p>")
</script>
<noscript>
  <p>Hola Mundo</p>
</noscript>
...
</body>
```

Si no se utilizan elementos NOSCRIPT en el caso de que los elementos SCRIPT afecten visualmente los documentos, es posible que algunos visitantes se pierdan de partes importantes de las páginas diseñadas.

El contenido del elemento NOSCRIPT solo se procesa y se muestra si el navegador Web del visitante no tiene soporte para *scripts*.

Capítulo 9. Evolución a XHTML

En este capítulo se estudia una introducción al lenguaje de programación XHTML, el sustituto natural de HTML como estándar para la elaboración de páginas Web.

9.1. ¿Qué es XHTML?

XHTML (Lenguaje Extendible de Marcado de Hipertexto) es un lenguaje de programación de páginas Web, al igual que HTML. Fue creado con la base de la especificación 4.01 de HTML integrando conceptos de XML (Lenguaje de marcado extendible), con la idea de ser el sustituto natural de HTML para escribir páginas Web. Básicamente, XHTML en su primera versión es "una reformulación de HTML en XML 1.0"[xhtml2000], cuya idea es poder crear documentos conformes con XML y a la vez interpretables por los agentes o navegadores Web para HTML 4.

En general, los beneficios que se obtienen al trabajar con XHTML son:

- Por ser conformes con XML, los documentos XHTML pueden ser legiblemente visualizados, editados y validados con herramientas para XML estándar.
- Los documentos en XHTML pueden funcionar tan bien o mejor que HTML en los navegadores Web diseñados para HTML 4 así como en los navegadores que soportan XHTML 1.0.
- Los documentos XHTML pueden incluir aplicaciones (*scripts* o *applets*) que hagan uso del modelo de objetos de dominio (DOM) bien sea de HTML o de XML.
- A medida que XHTML evolucione, los documentos van a ser cada vez más interoperables con otros ambientes.
- En XHTML es más fácil incluir nuevos elementos o atributos adicionales por medio de extensiones y módulos.

No todos los navegadores Web ofrecen soporte para páginas en XHTML. Entre los navegadores que no tienen soporte actualmente para XHTML se encuentra Microsoft Internet Explorer.[1]

Ejemplo 9.1. Estructura básica de un documento XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"❶>
<html xmlns="http://www.w3.org/1999/xhtml"❷>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8" ❸/>
  <title>Documento XHTML</title>
</head>

<body>

... Cuerpo del documento ...

</body>
</html>
```

Observaciones:

- ❶: Se debe cambiar la especificación de tipo de documento por la adecuada, dependiendo de si es de tipo *transitional*, *strict* o *frameset*, al igual que los documentos en HTML. En versiones posteriores de XHTML se elimina esta división.
- ❷: Se debe especificar al menos un espacio de nombres para las etiquetas utilizadas en el documento.
- ❸: El tipo de contenido para XHTML debería ser `application/xhtml+xml` en vez de `text/html`, pero se deja así para que los documentos XHTML versión 1.0 sean compatibles para la mayoría de navegadores Web. En versiones posteriores se espera cambiar ésto.

Se pueden guardar los archivos de código fuente de XHTML con estas extensiones de archivo: .xhtml, .xht, .html y .htm

9.2. Diferencias entre HTML y XHTML

A grandes rasgos las diferencias entre HTML y XHTML se listan a continuación:

- **Los documentos deben estar bien formados**. Bien formado quiere decir estructurado acorde con las reglas definidas por la recomendación de XML 1.0.

Ejemplo 9.2. Contraste de código mal formado y código bien formado para XHTML

Mal formado:

```
<p>Esto es un p&aacute;rrafo con texto en <em>cursiva</p></em>
```

Bien formado:

```
<p>Esto es un p&aacute;rrafo con texto en <em>cursiva</em></p>
```

- **Los nombres de elementos y atributos deben estar en minúscula**. En HTML se permite nombrar los elementos y atributos en mayúscula o minúscula sin diferencia alguna. En XHTML solamente se usan nombres en minúscula, ya que XML es sensible al formato de mayúsculas/minúsculas.

- **Se requiere etiqueta de cierre para elementos no-vacíos**. En HTML se permite omitir la etiqueta de cierre para algunos elementos, de forma que si se encuentra otra etiqueta de inicio, se da por hecho que la etiqueta que estaba abierta se ha cerrado. En XML, y por tanto en XHTML, no se permiten elementos con etiquetas de inicio y contenido sin etiqueta de cierre.

Ejemplo 9.3. Elementos no-vacíos en XHTML

Incorrecto:

```
<p>Primer p&aacute;rrafo
<p>Segundo p&aacute;rrafo
```

Correcto:

```
<p>Primer p&aacute;rrafo</p>
<p>Segundo p&aacute;rrafo</p>
```

- **Los valores de los atributos deben estar siempre entre comillas**. Todos los valores de los atributos deben estar encerrados por comillas dobles, sin importar si aparentemente son valores numéricos.

- **No se permiten atributos sin su respectivo valor**. XML no soporta la abreviación de atributos, es decir, especificar atributos sin valor, como el caso del atributo `multiple` en los elementos SELECT para indicar que se quiere obtener una lista de selección múltiple. En este caso se debe especificar el atributo y asignar algún valor, por ejemplo, `multiple="multiple"`.

- **Los elementos vacíos deben ser cerrados**. Elementos como BR o HR no tienen contenido, por lo que en HTML no hacía falta escribir una etiqueta de cierre como `</br>`, sino simplemente escribir la etiqueta de inicio `<br>`. Sin embargo, en XHTML es necesario cerrar los elementos vacíos utilizando una barra diagonal en la misma etiqueta de inicio o utilizando una etiqueta de cierre. Por ejemplo, en el caso del elemento BR se debería escribir `<br />` con un espacio antes de la barra diagonal o escribir `<br></br>`.

- **Se reducen los espacios en el valor de los atributos**. En HTML el valor de un atributo puede ser un texto con varios espacios al interior. Sin embargo, en XHTML este valor se reduce eliminando los espacios sobrantes al inicio y al final, además de que se reducen a uno solo los espacios que pueden haber como división entre una y otra palabra del texto.

- **El contenido de los elementos SCRIPT y STYLE es interpretado como parte del documento**. En XHTML el contenido de los elementos STYLE y SCRIPT es interpretado de forma diferente a como se hacía con HTML, de forma que si se encuentra un símbolo < o un símbolo & en el interior de estos elementos pueden considerarse como la indicación de inicio de una etiqueta (como <p>) o como la indicación de una entidad (como). Por esta razón, se recomienda tener el contenido de estos elementos en archivos aparte.

- **No se pueden especificar anidaciones prohibidas**. En HTML es posible especificar por medio del DTD algunas restricciones con respecto a elementos anidados. En XML y XHTML no es posible hacer ésto, de forma que se recomienda que se respeten estas prohibiciones a pesar de que no se puedan especificar. En general, estas prohiciones son:

  - Elementos A no pueden contener otros elementos A.

  - Elementos PRE no pueden contener elementos IMG, OBJECT, BIG, SMALL, SUB o SUP.

  - Elementos BUTTON no deben contener elementos INPUT, SELECT, TEXTAREA, LABEL, BUTTON, FORM, FIELDSET, IFRAME o ISINDEX.

  - Elementos LABEL no deben contener otros elementos LABEL.

  - Elementos FORM no deben contener otros elementos FORM.

- **Se suprime el atributo `name` y se utiliza únicamente el atributo `id` como identificador de fragmentos**. En HTML 4 se creó el atributo `id`, que al igual que el atributo `name` se utilizan para identificar fragmentos de código o elementos. En XHTML es obsoleto utilizar `name`, ya que solamente se requiere utilizar el atributo `id` para identificación.

- **Los conjuntos de valores predefinidos son sensibles al formato mayúsculas/minúsculas**. Atributos como `type` para los elementos INPUT ya tienen valores predefinidos, que en HTML se pueden escribir ya sea en mayúscula o minúscula sin alterar el resultado. Sin embargo, en XML y XHTML se tienen en cuenta las mayúsculas o minúsculas y se pueden considerr como dos valores diferentes, de forma que los valores predefinidos deben escribirse siempre en minúscula.

- **Las referencias hexadecimales a entidades son sensibles al formato mayúsculas/minúsculas**. En XML también se permite utilizar referencias a caracteres y símbolos utilizando su valor hexadecimal. Sin embargo, es diferente escribir `&#Xnn;` a `&#xnn;`, por lo que en XHTML siempre se deben escribir los valores hexadecimales en minúsculas.

Capítulo 10. Sitios Web

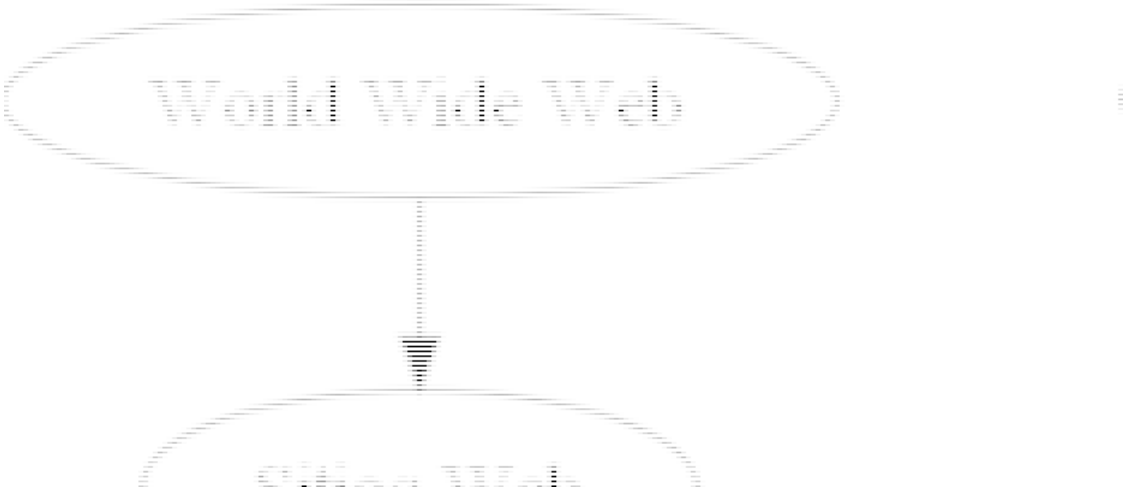
En este capítulo se explica cuál es la diferencia entre un sitio Web y una página Web, que son dos términos que se utilizan frecuentemente en Internet, y que a veces se utilizan como si fueran lo mismo, lo cual es un gran error. Luego veremos cómo se puede crear un sitio Web en Internet, y finalmente veremos cómo integrar diferentes tecnologías Web en sitios y páginas Web.

10.1. ¿Qué es un sitio Web?

10.1.1. Definición

Un sitio Web es una colección de páginas Web que relacionan texto, imágenes, videos o audio, presentando todo bajo un mismo dominio o subdominio en Internet. Una página Web es un documento escrito en HTML o una tecnología basada en HTML, al cual se accede por medio del protocolo HTTP (*HyperText Transfer Protocol*) utilizando un navegador Web como Mozilla Firefox. En este orden de ideas, el World Wide Web es una colección inmensa de sitios Web alojados en muchos servidores Web a lo largo y ancho del globo terráqueo.

Figura 10.1. Estructura del contenido del World Wide Web (WWW)



Usualmente los sitios Web tienen una página de inicio, que es básicamente la página Web principal del sitio.

Prácticamente todos los sitios Web visitables en Internet se alojan en un servidor Web, al cual se accede usualmente por medio de un registro en un sitio Web que preste el servicio de alojamiento. Este registro y el servicio de alojamiento puede ser gratuito, como bien puede ser pago. Ésto depende del proveedor del servicio y sus políticas, ya que incluso cuando el registro y el alojamiento son gratuitos, se le permite al proveedor incluir publicidad en los sitios Web que se alojen en esa modalidad.

10.1.2. Tipos de sitios Web

En Internet se puede encontrar gran variedad de sitios Web. Cada sitio Web tiende a pertenecer a una temática, que es la que determina su categoría(s). De acuerdo a la **temática** del sitio, se pueden encontrar en general las siguientes categorías de sitios Web (adaptado de Wikipedia):

- **Archivo**. Dedicados a almacenar y mostrar contenido antiguo de Internet.
- **Blog o Weblog**. Dedicado a la publicación de comentarios a diario o con cierta frecuencia.
- **Corporativo**. Sitios creados para la promoción comercial de una empresa y sus productos o servicios.

- **Compras**. Para la compra de bienes o servicios en Internet.
- **Comunidad virtual**. En estos sitios se reúnen personas con intereses comunes para compartir información de distinta índole.
- **Base de datos**. Sitios para consulta de datos especializados.
- **Desarrollo**. Sitios para la comunidad de desarrolladores de software y diseñadores gráficos.
- **Directorio**. Bases de datos especializadas en directorios de personas o empresas.
- **Juegos**. Dedicados a juegos de computador o consolas de videojuegos, y sus respectivos usuarios.
- **Descargas**. Dedicados a la descarga de software completo o en demostraciones y versiones de prueba.
- **Gubernamental**. Sitios creados por entidades del gobierno para la comunidad a la que apunta su trabajo.
- **Noticias**. Dedicados a la publicación de noticias de prensa utilizando medios audiovisuales y texto.
- **Pornografía**. Sitios para adultos con contenido explícito de tipo sexual.
- **Búsqueda**. Sitios para búsqueda de otros sitios o páginas de acuerdo a criterios concretos.
- **Informativo**. Para la publicación de información en general, por ejemplo, información de tipo turístico.
- **Espejo**. Sitio duplicado de uno original para servirle de respaldo.
- **Entretenimiento**. Sitios de humor, chismes, farándula y temas relacionados.
- **Personal**. Sitios creados por personas para autopublicitarse y darse a conocer en Internet.
- **Fraudulento**. Sitios creados con el fin de suplantar a otros sitios y robar la información de usuarios incautos.
- **Educativo**. Dedicados a la publicación de información para una comunidad académica y su entorno cercano.
- **Portal**. Un sitio que sirve como punto de inicio para el acceso a otros recursos en Internet.
- **Wiki**. Dedicados a compartir información creada de forma colaborativa.
- **Multimedia**. Dedicados a la publicación de contenido multimedial como videos y sonido en general (música, grabaciones, *podcasts*, etc.).

De acuerdo a la **forma** en que está construido y se presenta el contenido del sitio Web, se pueden clasificar los sitios Web en 1 de 2 grupos:

- **Estáticos**. El sitio se conforma por páginas Web y contenido estático, es decir, orientado a no cambiar con el tiempo a menos que sea editado y actualizado manualmente por su creador.
- **Dinámicos**. El sitio muestra contenido en páginas que se cargan de forma dinámica de acuerdo a las solicitudes del usuario, es decir, las páginas no están necesariamente almacenadas en un servidor Web de la forma en que se presentan, sino que se construyen al momento de ser consultadas.

10.2. ¿Cómo se crea un sitio Web?

Para crear un sitio Web se debe disponer de un servidor Web para alojar todo el contenido asociado. El acceso a este servidor Web depende del carácter del sitio, ya que puede que ya se cuente con un proveedor, si se trata de una empresa, o haya que buscar el proveedor si el sitio va a ser desarrollado y mantenido por una persona. También es necesario contar con un dominio o subdominio en Internet, para que el sitio tenga un nombre en Internet fácil de recordar y de encontrar en motores de búsqueda.

10.2.1. Alojamiento del contenido

Cada proveedor usualmente provee un cuadro comparativo de planes de alojamiento (o *hosting* en inglés) en sus servidores Web, de forma que se puede escoger la opción más adecuada de acuerdo a las necesidades de almacenamiento, tráfico, acceso a bases de datos, tecnologías Web adicionales y otros aspectos que pueda tener el sitio Web. Un ejemplo de ésto aparece en la Figura 10.2, “Planes de alojamiento para un proveedor en Internet (AwardSpace.com)”, para el caso del proveedor AwardSpace.com.

Figura 10.2. Planes de alojamiento para un proveedor en Internet (AwardSpace.com)

Una vez que se ha conseguido un proveedor de alojamiento y se ha hecho una solicitud de acuerdo al plan, se pueden comenzar a transferir archivos con el contenido de los sitios Web a los servidores en Internet, utilizando en la mayoría de los casos el protocolo FTP (*File Transfer Protocol* o Protocolo de Transferencia de Archivos) con unos datos de acceso que entrega el proveedor.

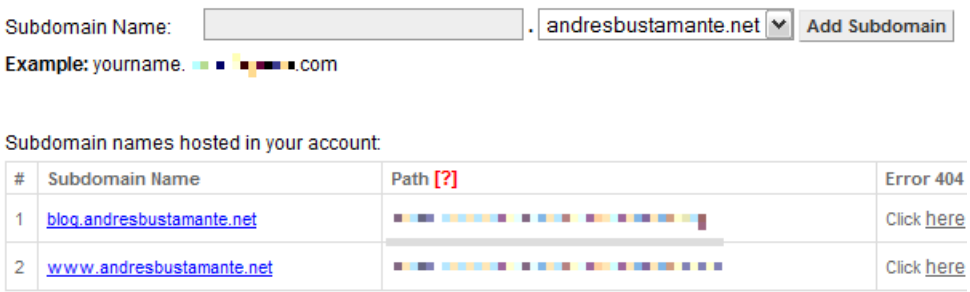
10.2.2. Creación del dominio o subdominio para el sitio

Es común que los mismos proveedores de alojamiento en Internet ofrezcan la posibilidad de crear dominios o subdominios para poder relacionar los contenidos alojados. Hemos visto que un sitio Web puede abarcar todo un dominio, como Yahoo.com como bien puede ser solamente un subdominio como blog.andresbustamante.net.

Si nuestro proveedor de alojamiento no nos da la facilidad de crear dominios o subdominios, tendremos que recurrir a proveedores de dominios como GoDaddy o 1and1, o buscar la manera de crear un subdominio que apunte a nuestros contenidos. Por el contrario, si nuestro proveedor de alojamiento nos da esta facilidad, podremos crear ya sea un dominio o subdominio (dependiendo del plan de alojamiento) para nuestro sitio Web utilizando el panel de administración que facilita el proveedor. El acceso a este panel debe estar descrito en los mensajes de correo electrónico que envía el proveedor al momento del primer registro.

Teniendo en cuenta la cantidad de proveedores que pueden existir, no es posible describir de forma única cómo crear un dominio o un subdominio en el panel de administración de un proveedor de alojamiento en Internet. Lo más común es encontrar un módulo de administración de dominios y subdominios, similar al que aparece en la Figura 10.3, “Panel de administración de subdominios en un proveedor de alojamiento de sitios Web”.

Figura 10.3. Panel de administración de subdominios en un proveedor de alojamiento de sitios Web



Si tuviéramos un proveedor como el que se utiliza para el ejemplo de la figura anterior, se observaría al frente de cada subdominio la dirección en el servidor donde se almacena el contenido para dicho subdominio, de forma que cada subdominio tiene contenido por aparte y así también cada sitio Web.

10.2.3. Creación del contenido del sitio Web

Para este punto vamos a suponer que se quiere publicar un sitio Web estático, es decir, sus páginas son diseñadas y luego almacenadas en un punto fijo en el servidor Web para que sean visualizadas de igual forma. Esta es la forma en la que se han venido trabajando las páginas Web de este curso, solo que no necesariamente son páginas que hablen de un solo tema o tengan relación entre si.

Mapa de navegación

Para crear el contenido de los sitios Web hay que crear primero un **mapa de navegación** para las páginas Web que va a contener el sitio, así como el contenido que se va a relacionar en las páginas (texto, video, audio, applets, flash, etc.). Este mapa no tiene un formato en particular, es simplemente saber cómo se van a relacionar las páginas entre si, partiendo de una página de inicio para el sitio. Muchas veces este mapa va a tener la forma de un árbol, donde la raíz del árbol es la página principal, y se van haciendo ramas a partir de cada página indicando otras páginas hacia las cuales apuntan.

Páginas Web del sitio

Luego de tener un mapa de navegación se elaboran las **páginas Web** como tal para el sitio, teniendo en cuenta aspectos de diseño del sitio, de forma que mantenga rasgos comunes no solo estéticos sino temáticos. Las páginas Web pueden estar hechas en HTML o XHTML, y opcionalmente apoyadas por lenguajes de programación como PHP, JSP, ASP u otro similar que permita ejecutar algunas líneas de código al momento de cargar las páginas.

Dependiendo del servidor Web donde se vayan a alojar las páginas puede que sea necesario renombrar los archivos, especialmente para las páginas de inicio del sitio y para las páginas principales de las carpetas interiores del sitio, ya que usualmente estas páginas deben tener nombres ya preestablecidos como `index.html` o `index.htm`.

Los sitios Web no son un conglomerado de páginas y contenido Web, sino que mantienen un orden y un estilo específico, de forma que se recomienda planear la organización de los archivos y ejecutar esta organización antes de poner el sitio en línea.

Publicación del contenido

Luego de tener las páginas Web, se procede a **publicar todo el contenido** de las carpetas locales con nuestras páginas y los archivos con contenido relacionado desde las páginas (imágenes, sonidos, podcasts, videos, etc.). Esta publicación se realiza habitualmente transfiriendo los archivos desde el computador donde se desarrolla el sitio hacia el servidor Web, utilizando programas especiales que soporten la transferencia por el protocolo FTP (*File Transfer Protocol*), teniendo en cuenta la ubicación de los archivos del sitio en el servidor, que se puede averiguar en el panel de administración de alojamiento de archivos de nuestro proveedor.

Al momento de incluir contenidos en las páginas Web como videos y audio, debe tenerse en cuenta que la ubicación de estos contenidos debe ser válida tanto para el momento de la edición de las páginas como para luego de ser publicadas, por tanto, deben revisarse las rutas a archivos invocados, incluyendo los que hacen referencia a *scripts* y hojas de estilo.

Una vez se hayan transferido los archivos con el contenido del sitio Web al espacio adecuado en el servidor, debe poderse observar el sitio ingresando a su subdominio o dominio asociado, según el caso, utilizando un navegador Web como Mozilla Firefox. Este mismo procedimiento se debe repetir cuando se quiere actualizar el contenido del sitio, ya sea actualizando la información de las páginas del sitio individualmente, o creando nuevas versiones del sitio para renovar su imagen.

10.3. Integración de tecnologías en sitios y páginas Web

Para integrar tecnologías como las que se mostraban en el esquema de la Figura 10.1, “Estructura del contenido del World Wide Web (WWW)” en sitios y páginas Web hay que seguir algunas recomendaciones:

- Incluir todos los contenidos de las tecnologías adicionales en las carpetas del sitio Web. Por ejemplo, mantener una carpeta para scripts en diferentes lenguajes, una para estilos, una para imágenes, una para sonidos y audio

en general, otra para videos, etc. La idea es mantener todo en las carpetas del sitio para no depender de otros sitios Web o terceros en general.

- Utilizar código HTML o XHTML **estándar**, de forma que los contenidos incluidos en las páginas sean visualizados igual en los navegadores Web más comunes. Esta recomendación no la siguen muchos programadores de páginas Web, que no tienen en cuenta que existen otros navegadores aparte de Internet Explorer, y que no tienen en cuenta que no todos los visitantes tienen el sentido de la visión en perfectas condiciones.

Se recomienda validar las páginas Web antes de publicarlas en Internet, utilizando un validador de HTML y XHTML como el que facilita el W3C en la dirección http://validator.w3.org/. Se puede validar una página ya publicada o se puede enviar el archivo de la página que no se ha publicado para verificar su validez y corregir a tiempo posibles errores.

El hecho de que una página sea mostrada por un navegador Web no es garantía de que la página esté bien elaborada y más aún, que cumpla con el estándar.

- Recordar que posiblemente los visitantes no tienen instalados los complementos necesarios para que el navegador Web visualice los contenidos que se agregan a las páginas Web. Es importante que estos complementos sean fáciles de conseguir y sobre todo, que funcionen en distintos sistemas operativos (p. ej. Adobe Flash Player). Se puede investigar en Internet si un archivo es reproducible en otros sistemas operativos diferentes de Windows.

Capítulo 11. Temas Avanzados

Más allá de HTML y su evolución XHTML existen otros temas relacionados que vale la pena revisar para terminar este curso. El primero tiene que ver con servidores Web, que como se comentaba en el capítulo anterior, son muy importantes para la publicación de sitios y páginas Web. El segundo tema avanzado tiene que ver con lenguajes de programación orientados a la Web, que funcionan como complemento a HTML especialmente para crear sitios Web dinámicos.

11.1. Servidores Web

11.1.1. ¿Qué es un servidor Web?

Un servidor Web es una aplicación de software que implementa el protocolo HTTP (*HyperText Transfer Protocol*), gracias a la cual se puede hacer la publicación de sitios y páginas Web a partir de archivos con código fuente escrito en lenguajes como HTML y XHTML. Esta aplicación se puede instalar en un computador convencional, así como en computadores dedicados, que es lo que se conoce como servidores en Internet. En cualquiera de los dos casos, la aplicación se ejecuta de forma permanente, de forma que se mantiene a la espera de solicitudes provenientes de clientes HTTP, es decir, navegadores Web.

11.1.2. Servidores Web más conocidos

Los sitios Web más conocidos para alojar páginas y sitios Web son los que se explican en esta sección.

Apache Web Server

El proyecto Apache publicó el servidor Web Apache como una aplicación de código abierto y gracias a ésto, a su confiabilidad y su portabilidad (se puede ejecutar en servidores Windows, Mac, Linux y UNIX en general) ha tenido una gran aceptación al punto de ser el servidor Web más importante en los últimos años.

Hoy en día este servidor hace parte de las plataformas LAMP (Linux, Apache, MySQL y PHP), muy utilizadas en Internet especialmente para sitios Web dinámicos que hacen uso de bases de datos y de alguna lógica especial programada en lenguaje PHP.

Internet Information Server (IIS)

Este producto desarrollado por Microsoft es la mayor competencia del servidor Web Apache, aunque no llega a ser tan aceptado, dado que funciona únicamente en servidores Windows (NT, 2000, XP, Server 2003, Server 2008), lo cual eleva significativamente el valor de su implementación por la vía legal, además de algunos problemas y vulnerabilidades encontrados tanto en el servidor Web como en el sistema operativo Windows.

Otros servidores Web

Además de los servidores Web ya mencionados, existen otros servidores alternativos, que se diferencian especialmente por su objetivo y por los lenguajes de programación que soportan para extender la funcionalidad de las páginas Web alojadas en él. A continuación se presenta un listado como las alternativas más utilizadas en la actualidad.

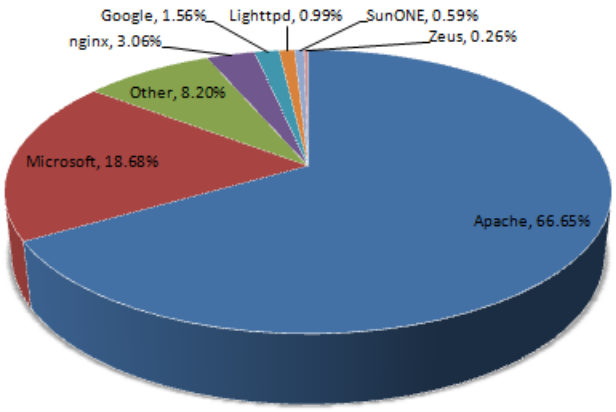
- QQ
- Sun Java System Web Server
- Apache Tomcat
- Google Web Server (GWS)
- nginx
- Zeus
- Mongrel

- Zope
- Pike
- Caudium
- Yaws
- Salvia
- Lighttpd

11.1.3. ¿Qué servidores Web utilizan los sitios más importantes?

Según estadísticas publicadas por la compañía Netcraft Ltd. en su sitio Web en Internet, el líder sobresaliente entre todos los servidores Web en Internet al mes de marzo de 2009 es el servidor Web Apache, con una participación de aproximadamente 66,6% de sitios Web en Internet. La muestra para esta estadística fue de un millón de sitios Web en Internet, correspondiente a los sitios Web más "ocupados" (*busiest*) o visitados en Internet.

Figura 11.1. Distribución de servidores Web entre el millón de sitios más visitados en Internet, a Marzo de 2009 (Tomado de Netcraft.com)



El servidor Web Apache ha mantenido su liderazgo entre los sitios Web en Internet desde el año de 1996, que a la fecha quiere decir que en promedio 2 de cada 3 sitios en Internet utilizan este servidor. Aquí está un pequeño listado de sitios Web clasificados por el servidor Web que utilizan:

- Servidor Web Apache: Facebook.com, Last.FM, Ask.com, AwardSpace.com, Terra.com, OpenOffice.org, Linux.org, HP.com, Flickr.com, Apple.com
- Internet Information Server: MSN.com, MySpace.com, Microsoft.com
- Google Web Server: Google.com.
- nginx: Wordpress.com
- No se puede identificar: Yahoo.com, Blogger.com, Geocities.com

11.2. Lenguajes de programación orientados a la Web

Esta sección explica lenguajes complementarios a HTML y XHTML para la programación de páginas Web. Los lenguajes acá explicados tienen la característica que se pueden mezclar en el código HTML o XHTML, y su ejecución se realiza en el servidor donde se encuentra alojado

11.2.1. PHP

PHP es un lenguaje de programación especial para la extensión de páginas Web dinámicas, y su nombre es un acrónimo recursivo que significa *PHP Hypertext Pre-processor*. Entre sus principales características está que es

un lenguaje desarrollado por una comunidad de desarrolladores, es libre y altamente portable (funciona en los sistemas operativos más conocidos), parcialmente orientado a objetos (desde la versión 5) y altamente extensible mediante módulos.

Ejemplo 11.1. Fragmento de código HTML con PHP (Original de Wikipedia)

```
<html>
<head>
    <title>Ejemplo de uso simple en envío y recepción de parámetros con PHP</title>
</head>
<body>
<?php
// Si existe la variable $_POST['muestra'], entonces muestra la comida favorita
if (isset($_POST['muestra']))
    {
        echo 'Hola, '.$_POST['nombre'].', tu comida favorita es:'.$_POST['comida'].'';
    } else {
// Si no, muestra un formulario solicitando la comida favorita
?>
<form method="POST" action="<?php echo $_SERVER['PHP_SELF'];?>">
    ¿Cuál es tu nombre?
    <input type="text" name="nombre" />
    ¿Cuál es tu comida favorita?
    <select name="comida">
        <option value="Spaguetis">Spaguetis</option>
        <option value="Asado">Asado</option>
        <option value="Pizza">Pizza</option>
    </select>
    <input type="submit" name="muestra" value="Seguir" />
</form>

<?php
    } //Fin del bloque else
?>

</body>
</html>
```

11.2.2. JavaServer Pages (JSP)

JavaServer Pages o JSP es una tecnología más que un lenguaje de programación, y es desarrollada por Sun Microsystems, al igual que el lenguaje de programación Java, sobre el cual se basa esta tecnología. Es una tecnología mucho más grande y posiblemente más robusta que PHP, dado el potencial que tiene Java como base, y a su facilidad para expresar acciones predefinidas y extenderse por medio de bibliotecas de etiquetas (*taglibs*).

Ejemplo 11.2. Fragmento de documento HTML con código JSP (Original de WIkipedia)

```
<%
   String titulo = "";
   if (request.getAttribute("titulo") != null) {
     titulo = (String) request.getAttribute ("titulo");
   }
 %>
 ...
 <title><%=titulo%></title>
 ....
```

JSP se utiliza en aplicaciones de gran exigencia y confiabilidad, y sobre todo cuando se trata de aplicaciones muy grandes que pueden requerir integración o comunicación con otras aplicaciones, especialmente a nivel empresarial.

11.2.3. ActiveServer Pages (ASP)

ASP es una tecnología desarrollada por Microsoft para su servidor Web Internet Information Server (IIS), para poder potenciar páginas Web en HTML con tecnologías de Microsoft como ActiveX y actualmente con la tecnología Microsoft .NET. Originalmente para escribir páginas con HTML y ASP se necesitaba tener nociones del lenguaje

de programación Visual Basic, pero ahora con ASP.NET es posible utilizar otros lenguajes del entorno .NET como C#, lo cual puede ser considerado como ventaja tanto como desventaja.

Ejemplo 11.3. Fragmento de código HTML con ASP (Original de Wikipedia)

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="1252"%>
(...)
<!--#include virtual="/Conexion a la base de datos.asp" -->
(...)
<p>Deja un Comentario
<%
  If (CStr(Recordset1.Fields.Item("ValorX").Value))=
     (CStr(Recordset2.Fields.Item("ValorY").Value))
  Then
%>
<strong><%=(Recordset2_total)%></strong>
<%
  else
%>
<strong>0</strong>
<%
  end if
%>
 Comentarios</p>
```

Al igual que JSP, es utilizado para aplicaciones grandes orientadas a la Web, con la diferencia de que está explícitamente orientada a ser implementada con y para tecnologías de Microsoft, como por ejemplo para el navegador Web Internet Explorer y el sistema operativo Windows.

Bibliografía

[ragget2005] RAGGET, Dave. Introduction to HTML. Disponible en línea: http://www.w3.org/MarkUp/Guide/.

[html1999] W3C. HTML 4.01 Specification. 1999. Disponible en línea: http://www.w3.org/TR/html4/.

[xhtml2000] W3C. XHTML 1.0 Specification. 2000. Disponible en línea: http://www.w3.org/TR/xhtml1/.

[wikipedia] Wikipedia, La enciclopedia libre. Disponible en línea: http://es.wikipedia.org/.