

XHTML+CSS  
de una maldita vez!  
(versión alpha 3)

Belén Albeza (BenKo)

12 de diciembre de 2004

# Índice general

<b>I</b>	<b>XHTML: La chicha</b>	<b>5</b>
<b>1.</b>	<b>Introducción al XHTML</b>	<b>6</b>
1.1.	¿Qué es el XHTML? . . . . .	6
1.2.	¿Y eso de CSS? . . . . .	7
1.3.	¿Cómo funciona el XHTML? . . . . .	7
1.4.	¿Pero para hacer webs no se usa el FrontPage? . . . . .	8
1.5.	Mi amigo que es diseñador usa Dreamweaver . . . . .	8
1.6.	Una preguntilla... . . . . .	8
<b>2.</b>	<b>Estructura de un documento XHTML</b>	<b>9</b>
2.1.	El DOCTYPE y la codificación . . . . .	9
2.2.	El elemento raíz (HTML) . . . . .	10
2.3.	La cabecera (HEAD) . . . . .	10
2.4.	El cuerpo (BODY) . . . . .	11
2.5.	Nuestra plantilla . . . . .	11
<b>3.</b>	<b>Etiquetas básicas</b>	<b>12</b>
3.1.	Párrafos . . . . .	12
3.2.	Saltos de línea . . . . .	12
3.3.	Títulos (headings) . . . . .	13
3.4.	Los Blockquotes . . . . .	13
3.5.	Separadores horizontales . . . . .	14
3.6.	Comentarios . . . . .	14
<b>4.</b>	<b>Etiquetas de formato</b>	<b>15</b>

4.1.	Énfasis . . . . .	15
4.2.	Preformato . . . . .	15
4.3.	Citas . . . . .	16
4.4.	Acrónimos y abreviaturas . . . . .	16
4.5.	Acerca de b y sus secuaces . . . . .	16
<b>5.</b>	<b>Enlaces</b>	<b>18</b>
5.1.	Enlace a una página externa . . . . .	18
5.2.	Enlace a una página local . . . . .	19
5.3.	Enlace a una dirección de e-mail . . . . .	19
5.4.	Anclas . . . . .	19
<b>6.</b>	<b>Listas</b>	<b>21</b>
6.1.	Listas ordenadas . . . . .	21
6.2.	Listas sin ordenar . . . . .	21
6.3.	Listas de definición . . . . .	22
6.4.	Listas anidadas . . . . .	22
<b>7.</b>	<b>Imágenes</b>	<b>24</b>
7.1.	Insertar una imagen . . . . .	24
7.2.	Imágenes como links . . . . .	25
7.3.	Sobre el uso y abuso de imágenes . . . . .	26
<b>8.</b>	<b>Tablas</b>	<b>27</b>
8.1.	Una tabla básica . . . . .	27
8.2.	Atributos de table . . . . .	28
8.3.	Expandir filas y columnas . . . . .	28
8.4.	¿Tablas para layouts? ¡Insensato! . . . . .	29
<b>9.</b>	<b>Formularios</b>	<b>31</b>
9.1.	La etiqueta FORM . . . . .	31
9.2.	Campos de texto . . . . .	32
9.3.	Campos de contraseña . . . . .	33
9.4.	Etiquetar campos . . . . .	33

9.5. Áreas de texto . . . . .	33
9.6. Casillas de verificación . . . . .	34
9.7. Botones de selección . . . . .	35
9.8. Listas de selección . . . . .	36
9.9. Botones de enviar y reestablecer . . . . .	36
<b>II CSS: La limoná</b>	<b>38</b>
<b>10.Introducción al CSS</b>	<b>39</b>
10.1. ¿Dónde escribo la parte de CSS? . . . . .	39
10.2. Vale, ¿pero cómo funciona? . . . . .	40
10.3. ¿Selectores? . . . . .	40
10.4. ¿Qué significa eso de Cascading? . . . . .	41
10.5. ¿Puedo poner comentarios? . . . . .	41
10.6. ¿Por qué fff significa blanco? . . . . .	41
<b>11.Trasteando por primera vez</b>	<b>43</b>
11.1. El color de primer plano . . . . .	43
11.2. El fondo . . . . .	43
11.3. Fuente . . . . .	44
11.4. Efectos en el texto . . . . .	45
<b>A. Migración rápida a XHTML</b>	<b>47</b>
A.1. Minúsculas y comillas, por favor . . . . .	47
A.2. Todas las etiquetas se cierran . . . . .	48
A.3. FONT y ciertos atributos desaparecen . . . . .	48
A.4. B y compañía también se van . . . . .	48
A.5. Hay que usar alt y title . . . . .	48
A.6. Cuidado al anidar etiquetas . . . . .	49
A.7. No existen los frames . . . . .	49
A.8. No se puede utilizar target . . . . .	49
A.9. Las tablas no se usan para maquetar . . . . .	50
A.10.Los & dan por saco . . . . .	50

<b>B. Licencia</b>	<b>51</b>
<b>C. Sobre esta versión</b>	<b>52</b>

Parte I

**XHTML: La chicha**

# Capítulo 1

## Introducción al XHTML

Este tutorial cubre cómo crear páginas web usando las tecnologías estándares recomendadas por el World Wide Web Consortium<sup>1</sup>: XHTML y CSS. No son necesarios conocimientos previos, así que lo único que necesitas es:

- Un **editor** de texto plano: Si usas *Windows*, sirve el *Bloc de Notas*. Ten cuidado si usas un procesador de textos, como el *Word*, ya que da formato y no queremos eso. Si usas *Linux*, pues tienes un montón para elegir: *vim*, *emacs*, *Joe*, *Kate*, etc.
- Un **navegador** que funcione **bien** y cumpla los **estándares**: el *Mozilla Firefox*, por ejemplo. Es multiplataforma, libre, y lo puedes bajar desde la web del proyecto *Mozilla*<sup>2</sup>.
- Un **navegador** que funcione **mal**, tenga todo el mundo y pase de los estándares: *Ya-sabes-cuál*<sup>3</sup>.

Es muy recomendable que tengas a mano el editor y un navegador para ir probando los ejemplos que salen aquí. Es la única forma de aprender.

### 1.1. ¿Qué es el XHTML?

**XHTML** significa *eXtensible HyperText Markup Language* y es la versión modernizada del tradicional HTML<sup>4</sup>, pero compatible con XML. Si ya conoces HTML, al final hay un apéndice para que sea más fácil la migración a XHTML.

Es un lenguaje **semántico**, lo que quiere decir que no definimos el *aspecto* de las cosas, sino lo que *significan*. Por ejemplo, si tenemos nuestro título de

---

<sup>1</sup>Es el organismo que se encarga de regular los estándares de la Web. Su URL es <http://www.w3c.org>

<sup>2</sup><http://www.mozilla.org>

<sup>3</sup>Su nombre está escrito en la Lengua Negra, que no pronunciaré aquí.

<sup>4</sup>Lenguaje utilizado para crear páginas web.

la página, en lugar de decir “*Lo quiero grande en letras rojas*”, le indicamos al navegador que “*Este es el título principal de la página. Haz algo para que destaque*”.

Obviamente, podemos controlar el aspecto que tienen nuestras páginas, pero eso es tarea de las CSS.

## 1.2. ¿Y eso de CSS?

**CSS** son las siglas de *Cascading Style Sheets* y son un regalo del cielo. Si el documento XHTML está bien estructurado, podemos cambiar totalmente su apariencia sin tocar una sola línea de código en el archivo .html. Esto nos permite **separar** el contenido del aspecto, y es de gran importancia.

Si queréis ver un ejemplo de cómo con el mismo código XHTML y distinto archivo CSS se pueden conseguir resultados muy diferentes, a la vez que espectaculares, echadle un vistazo a *CSS Zen Garden: The Beauty in CSS Design*<sup>5</sup>.

## 1.3. ¿Cómo funciona el XHTML?

XHTML está basado en **etiquetas**. Una etiqueta tiene la siguiente forma:

```
<etiqueta> Algo aquí dentro </etiqueta>
```

Volviendo al ejemplo anterior de nuestro título: la etiqueta para poner el título principal en la página es **<h1>**. Entonces nos quedaría:

```
<h1>What is the Matrix?</h1>
```

Como ves, **<h1>** marca el **inicio** de la etiqueta, mientras que **</h1>** se encarga de **cerrarla**. Hay etiquetas que funcionan con una sola parte, y son así:

```
<etiqueta />
```

Observa el espacio en blanco que hay entre el nombre de la etiqueta y la barra /. Es muy importante para que los navegadores antiguos no se vuelvan locos.

Hay etiquetas que pueden modificarse mediante **atributos**. Ahora mismo no hace falta entender qué hacen, ya los veremos más adelante. Sólo quédate con cómo van escritos:

```
<etiqueta atributo="valor">
```

---

<sup>5</sup><http://www.csszengarden.com>



Por último, comentar un par de reglas que siguen las etiquetas: siempre en **minúsculas** y los atributos **entre comillas**<sup>6</sup>.

## 1.4. ¿Pero para hacer webs no se usa el Front-Page?

Sí. También puedes freír huevos con aceite para el coche.

## 1.5. Mi amigo que es diseñador usa Dreamweaver

El *Macromedia Dreamweaver* es un editor WYSIWYG<sup>7</sup> muy extendido. Y muy bueno, por cierto. El problema es que nos permite hacer webs sin tocar nada de código.

Sí, eso es un problema. Genera código basura y no tenemos control sobre lo que hacemos. Así que antes de usar *Dreamweaver* o algo similar, tenemos que aprender a hacer las webs nosotros solos. Que quede claro que es mi opinión y no Ley Divina :P

## 1.6. Una preguntilla...

¿Has mirado primero en *Google*<sup>8</sup>? Es muy listo y lo sabe (casi) todo. Si aun así no te aclaras, mi e-mail es benko@ladybenko.net

---

<sup>6</sup>Esto es una diferencia respecto al HTML, ya que podíamos poner atributos sin comillas y no importaba si escribíamos en mayúsculas o minúsculas.

<sup>7</sup>What You See Is What You Get (Lo que ves es lo que obtienes).

<sup>8</sup><http://www.google.es>

## Capítulo 2

# Estructura de un documento XHTML

Este capítulo es tremendamente aburrido, pero muy importante. No es muy largo, así que presta atención. Ahora aprenderemos a formar el esqueleto de nuestros archivos para poder usarlo más adelante como plantilla.

### 2.1. El DOCTYPE y la codificación

La primera línea que debemos tener en nuestro documento XHTML es la que marca la **codificación**. ¿Qué es esto? Simplemente el formato en el que guardamos nuestro archivo. La codificación estándar es la **Unicode** (UTF-8) y soporta caracteres de todas las lenguas (occidentales, griegos, chinos, árabes, japoneses, coreanos...). Asegúrate de que el editor de textos que uses te guarda el archivo en formato Unicode<sup>1</sup>. Al grano, tenemos que escribir esto<sup>2</sup>:

```
<?xml version="1.0" encoding="UTF-8"?>
```

A continuación tenemos que indicar el **DOCTYPE**. Se encarga de decirle al navegador qué demonios contiene el archivo que está abriendo. Nosotros usaremos la especificación **XHTML 1.0 Strict**, que es esta:

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

Como ves, estas dos etiquetas no llevan la barra / de cierre. Son las únicas, así que no te malacostumbres. Por motivos de espacio, la línea del DOCTYPE

---

<sup>1</sup>Si usas *Bloc de Notas*, en el cuadro de diálogo de *Guardar Como...*, debajo del nombre del archivo aparece una lista desplegable para elegir la codificación.

<sup>2</sup>No somos masocas, así que no nos la tenemos que aprender de memoria. Simplemente copia y pega.

aparece cortada. No importa, porque el navegador interpreta los saltos de línea en el código como espacios en blanco. En realidad podríamos escribir todo el archivo XHTML en una sola línea. O cada palabra en una línea diferente.

Después del DOCTYPE tenemos a la cabecera y al cuerpo encerrados entre las etiquetas `<html>` y `</html>`

## 2.2. El elemento raíz (HTML)

El resto de nuestro documento tiene que ir encerrado por la etiqueta `<html>`. Pero en esa etiqueta tenemos que indicar una serie de cosas, como que nuestro documento es XHTML y la lengua que utilizamos. Si escribimos en castellano, nos quedaría así:

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

Las letras `es` son el código de la lengua correspondiente al castellano. Podéis ver algunos otros códigos<sup>3</sup> en el cuadro 2.1.

Cuadro 2.1: Algunos códigos de lenguas

Código	Lengua
es	Castellano
ca	Catalán
gl	Gallego
eu	Vasco
en	Inglés
fr	Francés
it	Italiano
de	Alemán
ja	Japonés

## 2.3. La cabecera (HEAD)

La cabecera contiene información que no forma parte del contenido de la página, como el título, vínculos a hojas de estilo CSS, información para robots de búsqueda, scripts, etc. Por ahora nos quedaremos sólo con el título de la página. La cabecera va encerrada entre `<head>` y `</head>` y el título entre `<title>` y `</title>`

```
<head>
```

---

<sup>3</sup>Los *trekkies* podéis escribir en vuestra lengua con el código x-klíngon.

```
<title>Título de la web</title>
</head>
```

El sangrado no es obligatorio, pero sí que viene muy bien para aclararnos mejor con el código.

## 2.4. El cuerpo (BODY)

Por último, nos encontramos con el cuerpo, encerrado entre `<body>` y `</body>`, y que contiene toda la “chicha”. Quedaría tal que así:

```
<body>
Aquí va el cuerpo de la web
</body>
```

## 2.5. Nuestra plantilla

Haciendo recopilación, nos quedaría algo como esto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

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

<head>
  <title>Título de la web</title>
</head>

<body>
Aquí va el cuerpo de la web
</body>

</html>
```

Y está pidiendo a gritos que guardes una copia en un archivito llamado *plantilla.html* para usos posteriores.

## Capítulo 3

# Etiquetas básicas

Ahora que conocemos la estructura de un documento XHTML, aprenderemos las **etiquetas básicas** que nos permitan empezar a crear el contenido nuestra página web: párrafos, saltos de línea, títulos, etc.

### 3.1. Párrafos

Los párrafos sirven para estructurar el contenido. En la web funcionan igual que en la vida real: contienen una o más frases con relación entre sí. Para crear un párrafo, metemos texto entre las etiquetas `<p>` y `</p>`. Un ejemplo<sup>1</sup>:

```
<p>
Hola, me llamo Luke Skywalker y soy piloto de una X-Wing, en el
Rogue Squadron. También soy un Jedi del Lado Luminoso de la Fuerza.
Mis maestros han sido Obi-Wan Kenobi y Yoda.
</p>
```

### 3.2. Saltos de línea

A veces, dentro de un mismo párrafo necesitamos cambiar de línea. Esto lo conseguimos con la etiqueta `<br />`, así:

```
<p>
Dark Chest of Wonders<br />
Seen through the eyes<br />
Of the one with pure heart<br />
Once, so long ago.
</p>
```

---

<sup>1</sup>Da igual insertar un salto de línea entre la etiqueta y el contenido, ya que será interpretado como un espacio en blanco.

Chulísima canción<sup>2</sup> de *Nightwish*, por cierto.

### 3.3. Títulos (headings)

Los títulos nos sirven para agrupar información. Imaginemos la sección de enlaces de nuestra página. El título principal podría ser *Mis Links Favoritos*. Luego tendríamos los enlaces clasificados en secciones (cada sección etiquetada con un subtítulo), como *Blogs*, *Descargas* y *Videojuegos*. Incluso en una de estas secciones podríamos tener varias categorías (marcadas con un sub-subtítulo), como *RPGs*, *Aventura*, *Acción* y *Lucha*.

Para conseguir esto, tenemos las etiquetas `<hx>` y `</hx>`, donde x es un número del 1 al 6. `<h1>` corresponde al título más importante y debería haber sólo uno por página. Veamos un ejemplo:

```
<h1>Mis aficiones</h1>
```

```
<h2>Los videojuegos</h2>
```

```
<p>
```

```
Mis videojuegos preferidos son los clásicos plataformas para la  
Master System y la Megadrive, como el Sonic, Alex Kidd, Dinamite  
Headdy, etc.
```

```
</p>
```

```
<p>
```

```
También me gustan las aventuras gráficas en 2D de toda la vida,  
como Goblins 3, Monkey Island, Woodruff, Simon the Sorcerer...
```

```
</p>
```

```
<h2>Música</h2>
```

```
<p>
```

```
Me gusta escuchar rock en inglés. Mis grupos favoritos son  
Dover, Nightwish, Sonata Arctica, HIM, etc. También escucho  
música en castellano, como Mecano, Mägo de Oz y La Oreja de Van  
Gogh.
```

```
</p>
```

### 3.4. Los Blockquotes

Los blockquotes sirven para mostrar un bloque que indica que lo que hay en el interior es una cita. Los navegadores normalmente los muestran con un indentado, aunque esto se puede cambiar con CSS. Nuestras canciones quedarían mejor así:

---

<sup>2</sup>*Dark Chest of Wonders*, de *Nightwish*, en el álbum *Once*.

```
<blockquote>
  <p>
    You think you have it all know:<br />
    The wisdom, power, know-how.<br />
    Can't even think you're wrong.<br />
    This is the way the brain of male is made.
  </p>
</blockquote>
```

Esta vez se trata de una de *Sonata Arctica*<sup>3</sup>.

### 3.5. Separadores horizontales

Los **separadores horizontales** (*horizontal rules*) han caído en desuso, ya que podemos conseguir bordes delimitadores mediante CSS. Pero como el saber no ocupa lugar, nos quedaremos con la etiqueta `<hr />` :

```
<h2>Los videojuegos</h2>
<p>Bla bla bla...</p>
```

```
<hr />
```

```
<h2>Música</h2>
<p>Bla bla bla</p>
```

### 3.6. Comentarios

Los **comentarios** son notas que ponemos en el código fuente de una página, pero que no se muestran. Para el navegador, son *invisibles*. Son útiles para indicar qué hacen ciertas partes del código. Para insertar un comentario, lo escribimos entre `<!--` y `-->` . Ten en cuenta que el comentario tiene que estar en una sola línea.

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

---

<sup>3</sup> *Weballergy*, de *Sonata Arctica*, en el álbum *Silence*

## Capítulo 4

# Etiquetas de formato

Ahora veremos cómo destacar ciertas partes del contenido de nuestra página para hacer más clara la lectura. Si ya tienes nociones de HTML, al final del capítulo hay un apartado para los aficionados de `<b>` y compañía<sup>1</sup>.

Hay que tener en cuenta que el aspecto de todos estos formatos son totalmente configurables con CSS.

### 4.1. Énfasis

Para dar **énfasis** a un texto disponemos de la etiqueta `<em>`. Si queremos dar un **énfasis más fuerte**, utilizaremos `<strong>`. Un ejemplo<sup>2</sup>:

```
<p>
<em>Far</em> is not the word because I'm never
far <strong>enough</strong>.
</p>
```

Los navegadores suelen representar `<em>` con cursiva y `<strong>` con negrita.

### 4.2. Preformato

Como ya os habréis dado cuenta, al interpretarse el código XHTML se ignoran espacios en blanco consecutivos, tabuladores, saltos de línea etc. Pero muchas veces nos interesa que estas cosas se tengan en cuenta (imagina si quieres insertar código fuente, por ejemplo). Para esto tenemos la etiqueta `<pre>`:

```
<pre>
```

---

<sup>1</sup>Recuerda: ¡son malvadas por naturaleza!

<sup>2</sup>De la canción *Far*, de *Dover*, en el álbum *Late at Night*



```
#include <stdio.h>

int main() {
    printf("Hola mundo!\n");
    return(0);
}
</pre>
```

### 4.3. Citas

Si queremos marcar un texto como una **cita**, utilizaremos `<cite>`. La diferencia con `<blockquote>` es que `<cite>` no crea la cita en un bloque, sino que la integra en el mismo texto. Los navegadores normalmente muestran el texto citado con cursivas.

```
<p>
Alguien escribió que <cite>la vida es sueño, y los sueños,
sueños son</cite>.
</p>
```

### 4.4. Acrónimos y abreviaturas

Para explicar el significado de los **acrónimos** y **abreviaturas** usaremos `<acronym>` y `<abbr>`, respectivamente. El efecto es el mismo: al pasar el ratón por encima de la palabra, nos aparece un rectángulo de ayuda con lo que significan.

Estas etiquetas necesitan un atributo para funcionar, y es `title`. Como valor de ese atributo escribiremos el texto con el significado del acrónimo o abreviatura. Así:

```
<p>
Mi ordenador tiene 512 <abbr title="Megabytes">Mb</abbr> de
memoria <acronym title="Random Access Memory">RAM</acronym>.
</p>
```

### 4.5. Acerca de b y sus secuaces

Si eres de la vieja escuela de webmasters, probablemente estés acostumbrado a dar formato al texto utilizando `<b>`, `<i>`, etc para conseguir texto en negrita, cursiva, subrayado, monoespaciado, etc.

Olvídate.

HTML 4 nos permitía indicar cómo iba a ser mostrado el texto. Por ejemplo, estaba la etiqueta `<font>`<sup>3</sup>, que nos permitía cambiar la fuente, tamaño y color del texto. Con XHTML nos olvidamos de eso y nos dedicamos a **estructurar** el texto y a darle un contenido **semántico**.

¿Qué es esto de la *semántica*? Deprisa y corriendo, viene a ser que una máquina entienda un texto. Un ordenador no sabe qué significa el que algo esté en negrita. Pero sí entiende que algo que tiene énfasis es importante, y debe ser destacado de alguna forma. Puede parecer una tontería, pero a la hora de trabajar con navegadores de texto, o para personas ciegas, o desde móviles, la cosa cobra importancia.

---

<sup>3</sup>La más malvada entre las malvadas.

# Capítulo 5

## Enlaces

Ya sabemos las etiquetas necesarias para escribir texto. Ahora toca aprender a usar una de las características más importantes de la web: los enlaces (o links). Se implementan con la etiqueta `<a>`.

### 5.1. Enlace a una página externa

Si queremos un link a una página que esté en **otro servidor**, usamos `<a>` de esta forma:

```
<a href="http://www.loquesea.com" title="Descripción">
Texto del enlace
</a>
```

El atributo `href` contiene la **URL** a donde queremos enlazar. Es muy importante que no se nos olvide el `http://` o no funcionará.

En `title` escribimos una **descripción** del sitio web al que enlazamos. Al igual que con las etiquetas `<abbr>` y `<acronym>`, el texto aparecerá al pasar el ratón por encima del link. No hay que confundir el título con el texto del enlace. Son completamente independientes.

Imagina que quieres enlazar a Barrapunto<sup>1</sup>:

```
<a href="http://www.barrapunto.com" title="Barrapunto:
la información que te interesa">Barrapunto</a>
```

---

<sup>1</sup>Blog colectivo geek, similar a Slashdot. <http://www.barrapunto.com>

## 5.2. Enlace a una página local

Para enlazar a una página que esté en **nuestro servidor**, necesitamos saber la **ruta** (*path*) desde donde estemos hasta la ubicación del archivo.

Si el origen (página donde está el link) está en el **mismo directorio** que el destino (página a la que apunta el link), entonces sólo tenemos que escribir su nombre:

```
<a href="profile.html" title="Información sobre mí">
Ficha personal
</a>
```

Si el destino estuviera en un subdirectorio, utilizamos una barra / para indicar el camino<sup>2</sup>:

```
<a href="galeria/color.html" title="Galería color">
Ver dibujos a color
</a>
```

Si el destino estuviera un directorio por encima, se usan dos puntitos y una barra ../ de esta manera:

```
<a href="../index.html" title="Página principal">
Volver al inicio
</a>
```

## 5.3. Enlace a una dirección de e-mail

Podemos hacer un enlace que al pulsar sobre él, se abra automáticamente una ventana del cliente de correo electrónico que tenga el usuario<sup>3</sup> para que escriba un mensaje a esa dirección.

Sólo tenemos que poner en href la palabra **mailto:** seguida de una dirección de correo electrónico. Así:

```
<a href="mailto:leia@alianza.net"
title="E-mail de la princesa Leia">Leia</a>
```

## 5.4. Anclas

Las anclas nos permiten enlazar a una posición concreta de la página, en plan teletransporte. Funcionan así:

---

<sup>2</sup>En Windows para mostrar una ruta, se utilizan las barras invertidas \ para separar los directorios. Aquí usamos las barras normales.

<sup>3</sup>Normalmente el infame Outlook. Usa Thunderbird (<http://www.mozilla.org>), que es mejor y libre.

En el punto al que queremos saltar, insertamos un ancla (usamos el atributo `id`<sup>4</sup>):

```
<a id="nuestra_ancla" />
```

Si ahora vemos la página en el navegador, no notaremos ninguna diferencia, puesto que las anclas son invisibles. Para usarlas, debemos crear un link que nos lleve a ese ancla:

```
<a href="#nuestra_ancla" title="descripción">Texto del enlace</a>
```

Observa que en el enlace hay que poner una almohadilla `#` delante del nombre del ancla. Cuando creamos el ancla, no es necesaria la almohadilla.

También podemos enlazar a un ancla que esté en otra página:

```
<a href="otra_pagina.html#ancla" title="descripción">
Texto del enlace
</a>
```

Las anclas son muy útiles para páginas muy extensas. Un ejemplo lo tenemos en las FAQ<sup>5</sup>. Las FAQ tienen un índice de preguntas que enlazan al ancla correspondiente. Si hacemos clic en la pregunta número 3, nos enlazará a ese punto de la página. Normalmente las preguntas tienen otro enlace que nos devuelve a un ancla situada en el índice de preguntas, para facilitar la navegación.

---

<sup>4</sup>En HTML 4 se utilizaba el atributo `name`, pero con XHTML usamos `id`.

<sup>5</sup>Recopilación de las Preguntas Más Frecuentes sobre un tema, contestadas.

# Capítulo 6

## Listas

Ahora veremos como implementar **listas** en nuestras páginas. Las hay de tres tipos: ordenadas, sin ordenar y de definición. Venga, que esto es muy facilito.

### 6.1. Listas ordenadas

Las **listas ordenadas** muestran sus elementos numerados. Se crean con la etiqueta `<ol>`:

```
<p>Mis escritores favoritos (en orden de preferencia):</p>
<ol>
  <li>R.A.Salvatore</li>
  <li>J.K.Rowling</li>
  <li>Isabel Allende</li>
</ol>
```

Hay que tener en cuenta que con CSS podemos cambiar el número por el que queremos empezar a contar, así como el tipo de numeración (números arábigos, romanos, letras del abecedario, mayúsculas...).

### 6.2. Listas sin ordenar

Las **listas sin ordenar** marcan a cada elemento con una viñeta, de modo que no se sigue una numeración. Se crean con la etiqueta `<ul>`:

```
<p>El helado perfecto (ñam!):</p>
<ul>
  <li>1 bola de helado de chocolate</li>
```

```
<li>1 bola de helado de limón</li>
<li>Trocitos de piña y melocotón en almíbar</li>
<li>Sirope de chocolate</li>
</ul>
```

Al igual que con las listas ordenadas, podemos modificar su apariencia con CSS y elegir el tipo de viñeta que queremos.

Por cierto, prueba el helado, está riquísimo<sup>1</sup>.

### 6.3. Listas de definición

Las **listas de definición** se diferencian de las anteriores en que cada “ítem” está compuesto por un par de elementos: un **término** y su **definición**. Se usan estas etiquetas: `<dl>` para marcar la lista, `<dt>` para un término y `<dd>` para su definición.

```
<p>Significado de algunos smileys:</p>
```

```
<dl>
  <dt>:</dt>
  <dd>Sonrisa</dd>

  <dt>xD</dt>
  <dd>Carcajada</dd>

  <dt>P</dt>
  <dd>Sacar la lengua</dd>
</dl>
```

Y sí, también se puede cambiar el aspecto y bla, bla, bla. Comentar que las palabras *término* y *definición* no sólo se refieren a algo con su significado, también podemos usar una lista de definición para crear un *profile* (por ejemplo), relacionando los pares *Nombre-Leía*, *Ciudad-Coruscant*, *Profesión-Senadora*.

### 6.4. Listas anidadas

No, las **listas anidadas** no son un nuevo tipo de listas, sólo una combinación de las anteriores. *Anidar* significa meter una lista dentro de otra. Por ejemplo:

```
<ul>
  <li>I Trilogía del Elfo Oscuro</li>
  <li><ol>
    <li>La Morada</li>
```

---

<sup>1</sup>También puedes añadir guindas y nata.

```
    <li>El Exilio</li>
    <li>El Refugio</li>
  </ol></li>
  <li>Trilogía del Valle del Viento Helado</li>
</ul>
```

No es difícil, sólo debemos tener cuidado cerrando la etiqueta que toque. ¿Cómo lo sabemos? Fácil: **primero se cierran las interiores**, y después las de fuera. Así, si tenemos <li><ol> tenemos que poner primero </ol> y después </li>.



# Capítulo 7

## Imágenes

Las imágenes son un elemento importante a la hora de hacer más atractiva una web. No obstante, hay que saber cuándo utilizarlas y no abusar de ellas.

Podemos usar tres formatos de imagen: GIF, JPEG y PNG. El JPEG es el más adecuado para imágenes con muchos colores, como **fotografías**, y que no tengan grandes áreas de colores planos. Las imágenes GIF son de sólo 8 bits (256 colores), pero podemos usar un color **transparente**. El formato PNG es el estándar y podemos elegir varias profundidades de paleta (número de colores). Además, podemos utilizar un **canal alpha** para crear transparencias. Pero Ya-Sabéis-Cuál navegador no implementa correctamente los PNG, así que hay que llevar cuidado.

### 7.1. Insertar una imagen

Para poner una imagen hacemos uso de la etiqueta `<img>` con unos cuantos atributos:

```

```

Con `src` establecemos qué imagen queremos mostrar. Al igual que con los links, hay que tener en cuenta la **ruta** hacia la imagen. Por motivos de organización, normalmente tendremos las imágenes en un subdirectorio (o en varios) llamado `images`, así que pondríamos `src="images/algo.gif"`.

Los atributos `width` y `height` nos permiten establecer el **ancho** y el **alto** de la imagen. Podemos indicar un valor en píxeles o en tanto por ciento. `width="200"` muestra la imagen con 200 píxeles de ancho, mientras que `width="200%"` hace que se vea al doble de su anchura original. Estos dos atributos no son obligatorios, pero sí es conveniente que pongamos las dimensiones en píxeles reales, ya que ahorramos tiempo al navegador a la hora de maquetar la página.

El atributo `alt` contiene una **descripción** de la imagen, que veremos cuando pasemos el ratón por encima y mientras se carga el fichero. También se muestra esta descripción cuando no se ha podido cargar la imagen. Es un atributo **obligatorio**, por cuestiones de accesibilidad. Hay personas que deshabilitan la carga de imágenes para ahorrar tiempo. Otras usan navegadores de texto como el *lynx*. Y otras, sencillamente son ciegas.

Como ejemplo, así quedaría una imagen de banner:

```

```

## 7.2. Imágenes como links

También podemos hacer que una imagen sea a su vez un **enlace** a una página. Los navegadores suelen mostrarla con un reborde para indicarnos que se trata de un link, pero lo podemos cambiar con CSS.

Para poner una imagen como un link, la introducimos dentro de la etiqueta `<a>`:

```
<a href="http://www.ladybenko.net" title="Portafolio online">

</a>
```

Hay una técnica a la hora de implementar **galerías de imágenes**, y es hacerlo con *thumbnails*. ¿Qué es esto? Pues una *thumbnail* es una imagen más pequeña que la original, de modo que al hacer clic sobre ella, cargamos la imagen a tamaño completo. Entonces, algunos iluminados hacen:

```
<a href="matrix.jpg" title="Wallpaper de Trinity">

</a>
```

¡Olé campeones! Si nuestro wallpaper de Trinity ocupa 100Kb (o más), tendremos esos 100Kb ¡como thumbnail! (justo lo que queremos evitar). El escalar una imagen con `width` y `height` no hace que ocupe menos espacio<sup>1</sup>. Tenemos que coger un programa de dibujo, escalar la imagen y guardar esta copia más pequeña (de 5Kb, por ejemplo):

```
<a href="matrix.jpg" title="Wallpaper de Trinity">

</a>
```

---

<sup>1</sup>Firefox no es *tan* bueno ;)

### 7.3. Sobre el uso y abuso de imágenes

Se dice que una imagen vale más que mil palabras. Cierto, pero muchas imágenes, o pocas mal puestas, pueden llegar a desesperar.

¿Te resulta esto familiar? Entrás a una web con un fondo de color chillón, letras verde fosforito, una cantidad ingente de GIFs animados, marquesinas, applets de Java, etc<sup>2</sup>. ¿Cuánto tiempo tardas en cerrar la página? Yo es que ya actúo por reflejo.

Es por esto que debemos limitar los GIFs animados al máximo, así como evitar el uso indiscriminado de imágenes. Recuerda: **sólo hay que poner las imágenes necesarias.**

---

<sup>2</sup>Aunque parezca mentira, esto estaba muy de moda en los tiempos de los módems a 56K.

# Capítulo 8

## Tablas

Las **tablas** son el mecanismo que nos proporciona XHTML para presentar **información tabulada**, como horarios o la clasificación de la Liga. Son un poco engorrosas de utilizar, pero a veces son necesarias. Así que allá vamos.

### 8.1. Una tabla básica

Disponemos de las siguientes etiquetas para crear una **tabla**:

- `<table>`: Crea la tabla
- `<caption>`: Pone título a la tabla
- `<tr>`: Crea una fila
- `<td>`: Crea una celda
- `<th>`: Crea una celda de encabezamiento

Se entiende mejor con un ejemplo. Es muy conveniente utilizar bien los sangrados, ya que hay que tener mucho cuidado con cerrar cuando corresponda las etiquetas `<tr>`, `<td>` y `<th>`. Aquí tenemos a una tabla de 2x2:

```
<table>
  <caption>Videojuegos</caption>

  <tr>
    <th>Titulo</th>
    <th>Género</th>
  </tr>
  <tr>
    <td>Sonic</td>
    <td>Plataformas</td>
```

```
</tr>
</table>
```

El mecanismo es siempre el mismo. Por cada fila que queramos, abrimos una etiqueta `<tr>` e insertamos allí las celdas que correspondan. Dentro de cada celda podemos insertar prácticamente **cualquier cosa**, pero debemos tener siempre en mente que el objetivo de las tablas es tabular información.

## 8.2. Atributos de table

La etiqueta `<table>` dispone de una serie de **atributos**<sup>1</sup> que nos permiten modificar su borde y los márgenes de las celdas.

Para cambiar el tamaño del borde de la tabla, usamos `border` con un valor en píxeles. Si no indicamos nada, los navegadores suelen tomar como valor por defecto 1 ó 0. Si no queremos ningún borde, debemos utilizar `border="0"`.

Si lo que queremos es cambiar la distancia entre una celda y otra, empleamos el atributo `cellspacing` con un valor en píxeles. Y para modificar la distancia del contenido de la celda a los bordes de esta, usamos `cellpadding`<sup>2</sup>. La diferencia entre `cellspacing` y `cellpadding` puede confundir al principio, así que lo mejor es verlo con un ejemplo<sup>3</sup> (modificamos la tabla anterior):

```
<table border="1" cellpadding="10" cellspacing="30">
  <caption>Videojuegos</caption>

  <tr>
    <th>Titulo</th>
    <th>Género</th>
  </tr>
  <tr>
    <td>Sonic</td>
    <td>Plataformas</td>
  </tr>
</table>
```

## 8.3. Expandir filas y columnas

Muchas veces necesitamos que una celda ocupe más de un espacio. Pongamos como ejemplo nuestra socorrida tabla de videojuegos. ¿Qué pasa si queremos meter más de un género de cada videojuego? Podemos hacer que la celda de género ocupe dos columnas y así podremos introducir dos géneros por cada juego. Así:

---

<sup>1</sup>Los viejos atributos que en HTML 4 nos permitían establecer anchuras y colores son sustituidos por CSS.

<sup>2</sup>Aunque, en mi opinión, es mejor usar CSS para esto.

<sup>3</sup>Los valores están muy exagerados para que se note la diferencia.

```

<table>
  <caption>Videojuegos</caption>

  <tr>
    <th>Titulo</th>
    <th colspan="2">Géneros</th>
  </tr>
  <tr>
    <td>Sim City</td>
    <td>Simulación</td>
    <td>Estrategia</td>
  </tr>
</table>

```

Como ves, para expandir una celda varias columnas hemos usado el atributo `colspan`. Podemos hacer lo mismo con las filas, utilizando `rowspan`. Vamos a hacer la misma tabla de antes, pero girada 90 grados:

```

<table>
  <caption>Videojuegos</caption>
  <tr>
    <th>Titulo</th>
    <td>Sim City</td>
  </tr>
  <tr>
    <th rowspan="2">Géneros</th>
    <td>Simulación</td>
  </tr>
  <tr>
    <td>Estrategia</td>
  </tr>
</table>

```

¿Qué es un rollo esto de las tablas? Pues sí, pero no le des mucha importancia. La mayoría de editores de código HTML, como el *Quanta* o el *Homesite +*, traen asistentes para crear tablas de una forma más rápida y sencilla.

## 8.4. ¿Tablas para layouts? ¡Insensato!

A día de hoy, la mayoría de las páginas web están maquetadas usando tablas con `border="0"`. Antes de la llegada de CSS era totalmente imposible crear texto a columnas y, en definitiva, maquetar un sitio web<sup>4</sup>. Afortunadamente, CSS implementa **capas**, con lo que se puede configurar totalmente la apariencia y colocación de cada elemento de la página mediante la hoja de estilos, dejando el código XHTML de forma muy sencilla.

<sup>4</sup>Bueno, con *frames* se podía hacer algo, pero era peor el remedio que la enfermedad.

¿Entonces por qué la gente sigue usando tablas? Porque piensan que eso de las capas es algo muy complicado. ¡Mentira! Lo que pasa es que nadie nace enseñado y aprender algo nuevo siempre da un poco de pereza. En la parte de CSS del tutorial aprenderemos a crear varios tipos de diseño populares para así tumbar el mito de que las capas son difíciles.

Y, dándole la vuelta a la tortilla, **¿por qué no usar tablas?** Pues porque las tablas no se han creado para maquetar y el WWW Consortium lo desaconseja. Además, en navegadores no visuales (de texto, para ciegos o otros dispositivos que no sean un monitor) el resultado es totalmente **imprevisible**. ¿Quieres otra razón de peso? Las tablas dan más trabajo. El código queda más enrevesado y si queremos renovar el diseño de nuestra página **hemos de cambiar prácticamente todo**, mientras que si usáramos un layout por capas sólo tendríamos que modificar el CSS. ¿Más razones? Al ser el código más complicado, las páginas son **más pesadas** y gastan más ancho de banda. Además de que **tardan más en cargar**, ya que hasta que no se carga todo lo que hay en la tabla, no se muestra el resultado.

El futuro es XHTML y sitios web importantes, como Blogger<sup>5</sup> ya han rediseñado su web con capas. Así que te animo a que cumplas los estándares y no uses las tablas para maquetar<sup>6</sup> :)

---

<sup>5</sup>Comunidad de Blogs de Google <http://www.blogger.com>

<sup>6</sup>En la web de *Steal these buttons!* hay botoncitos muy cucos para layouts *tableless* (sin tablas).

## Capítulo 9

# Formularios

Los formularios nos permiten recoger información introducida por el usuario. Esta información podemos enviarla por correo electrónico o procesarla con un script.

Si se manda por correo electrónico, hay que tener en cuenta que es información no cifrada y que podría ser interceptada, así que no debe contener datos importantes. Un uso aceptable sería un formulario con comentarios sobre nuestra página o para pedir un intercambio de links.

Si nuestro servidor dispone de tecnología como PHP o CGI (por ejemplo), podemos hacer más cosas con esa información, como guardarla en una base de datos o generar una página dinámicamente.

### 9.1. La etiqueta FORM

Todo formulario está encerrado por `<form>` y `</form>`. Dentro de estas etiquetas, van los campos del formulario, y podemos usar párrafos y saltos de línea. Vamos a ver un ejemplo de etiqueta `<form>`, suponiendo que va a ser un formulario que se envía por correo electrónico:

```
<form action="mailto:leia@alianza.net" method="post"
enctype="text/plain">
```

El atributo `action` recoge qué se va a hacer una vez que se pulse el botón de enviar. En ese ejemplo, el formulario se envía a la dirección `leia@alianza.net`. Si tuviésemos un script para procesar el formulario, hubiésemos puesto algo como `action="enviar_info.php"`.

Con `method` especificamos cómo va a ser enviada la información. Si utilizamos correo electrónico, le damos el valor de `post`. Para scripts se suele uti-



lizar `method="get"`, que pone el valor de las variables en la misma URL<sup>1</sup>.

Por último, con `enctype="text/plain"` conseguimos que a nuestro buzón llegue el formulario en texto plano sin caracteres extraños.

## 9.2. Campos de texto

La mayoría de los campos de un formulario se crean con una sola etiqueta, `<input>` y mediante su parámetro `type` especificamos el tipo de campo que queremos. Un **campo de texto** básico quedaría así:

```
<input type="text" id="nombre" name="nombre" size="20" />
```

Veamos ahora los atributos. Con `type="text"` indicamos que se trata de un campo de texto. El atributo `size` recoge el ancho del campo, medido en caracteres. Ahora bien, ¿qué hacen `id` y `name`?

Pues `id` es un **identificador**. Esto implica que nada en todo el documento puede llamarse del mismo modo. Este parámetro sirve para CSS<sup>2</sup> y también para otras cosas como su uso con la etiqueta `<label>` (que veremos a continuación).

Con `name` damos **nombre a la variable** de ese campo. Por ejemplo, si el usuario escribe “Morpheo” en el campo que hemos puesto antes de ejemplo, y lo envía, recibiríamos en nuestra bandeja de entrada un mail que contendría algo así:

```
nombre=Morpheo
```

De todos modos, para ahorrarnos problemas, siempre que podamos es mejor escribir el mismo valor para `id` y `name`. Hay que complicarse la vida lo menos posible :)

Hay otros atributos adicionales para nuestros campos de texto. Podemos indicar un **número máximo de caracteres** que puede introducir el usuario con `maxlength`. Y si queremos que aparezca un **valor por defecto**, utilizamos `value="algo"`. Por ejemplo, si queremos pedir la dirección de su página web:

```
<input type="text" name="url" id="url" size="30"
maxlength="255" value="http://" />
```

Y, por supuesto, no debemos olvidarnos de nuestro tan socorrido `title`, que funciona igual que con la etiqueta `<a>`.

---

<sup>1</sup>Si todo esto te suena a chino, no te preocupes. Sólo necesitas saberlo si utilizás un script... Y si lo haces, ya te enseñarán cuando aprendas a escribirlos ;)

<sup>2</sup>Estableciendo un aspecto exclusivo para ese elemento, por ejemplo.

### 9.3. Campos de contraseña

Los **campos de contraseña** son exactamente iguales que los de texto, sólo que el usuario en lugar de ver los caracteres que ha introducido, ve asteriscos. Lo de exactamente iguales quiere decir que es texto, la información no va cifrada de ninguna manera. La diferencia entre un campo de texto y uno de contraseña es meramente estética.

Los atributos son los mismos que con los campos de texto, lo único que cambia es que debemos introducir `type="password"`. Ejemplillo:

```
<input type="password" name="pass" id="pass" size="20" />
```

### 9.4. Etiquetar campos

Ya hemos aprendido a crear campos de texto para nuestro formulario, ¿pero cómo le decimos al visitante qué es lo que tiene que introducir en cada campo? Podríamos hacer algo así:

```
<p>
Nombre: <input type="text" name="nombre" id="nombre" size="20" />
<br />
E-mail: <input type="text" name="mail" id="mail" size="40" />
</p>
```

Sin embargo, podemos tener problemas en navegadores no visuales. ¿Cómo sabemos que la palabra “nombre” hace referencia al campo con el atributo `id="nombre"`? Para eso disponemos de una etiqueta nueva, `<label>`.

Esta etiqueta se encarga de asociar texto con su campo de formulario correspondiente. Sólo tiene un atributo, `for`, y en el tenemos que indicar la `id` del campo al que queremos hacer referencia. Nuestro ejemplo anterior sería más correcto así:

```
<p>
<label for="nombre">Nombre:</label>
<input type="text" name="nombre" id="nombre" size="20" />
<br />
<label for="mail">E-mail:</label>
<input type="text" name="mail" id="mail" size="40" />
</p>
```

### 9.5. Áreas de texto

Con las **áreas de texto** damos al usuario la posibilidad de introducir texto con varias líneas. La etiqueta a usar es `<textarea>` y su funcionamiento es algo diferente al de `<input>`.

La etiqueta `<textarea>` dispone de los atributos `id`, `name` y `title`, y funcionan como en el resto de campos de formulario. Además, disponemos de otros dos para indicar las dimensiones del área de texto: `cols` se encarga de establecer el ancho (medido en caracteres) y `rows` el alto, medido en líneas.

Como ejemplo, imaginemos que en una parte del formulario queremos poner un campo para que el usuario deje un comentario. Como probablemente sea largo, utilizamos un `textarea`:

```
<p>
<label for="comentario">Comentario:</label><br />
<textarea name="comentario" id="comentario" cols="30" rows="5">
Bla bla bla
</textarea>
</p>
```

Fíjate en que `<textarea>` tiene etiqueta de cierre. El texto que hay entre la etiqueta de apertura y la de cierre es el valor por defecto que contendrá el campo, en este caso `Bla bla bla`.

## 9.6. Casillas de verificación

Una **casilla de verificación** (más conocida como *checkbox*) es un cuadrado que el usuario puede activar y desactivar pulsando en él. Se crean con la etiqueta `<input>` y escribiendo `type="checkbox"`. Los atributos `id`, `name` y `title` funcionan con normalidad, pero `value` se usa de forma algo diferente aquí. Lo que escribamos en `value` es lo que nos saldrá en el mail que recibamos como el valor de la variable (indicada en `name`) si la casilla está activada. Es decir, que si ponemos esto...

```
<p>
Has jugado a...<br />
<input type="checkbox" name="monkey1" id="monkey1" value="si" />
<label for="monkey1">Monkey Island I</label>
</p>
```

... y el usuario activa la casilla, recibiremos un mail tal que así:

```
monkey1=si
```

También podemos hacer que una casilla esté activada por defecto, si añadimos el atributo `checked="checked"`. Así:

```
<p>
Has jugado a...<br />
```

```

<input type="checkbox" name="monkey1" id="monkey1" value="si"
checked="checked" />
<label for="monkey1">Monkey Island I</label>

<input type="checkbox" name="xwing" id="xwing" value="si" />
<label for="xwing">X-Wing Alliance</label><br />
</p>

```

Nos quedaría la casilla etiquetada como *Monkey Island*<sup>3</sup> activada, mientras que la de *X-Wing Alliance* sin activar.

## 9.7. Botones de selección

No sé si el término *botones de selección* es en realidad el más adecuado como traducción, así que mejor una explicación de lo que hacen. El nombre en inglés es **radio button** y son casillas circulares agrupadas en las que sólo una puede estar activada. Sirven para cuando queremos que el usuario sólo seleccione una opción de las múltiples que se le dan.

Aunque se crean con la etiqueta `<input>` indicando `type="radio"`, los radio button son algo “especiales”, así que los veremos un poco más despacio.

Pongamos el caso de que queremos que el visitante de nuestra web nos indique cuál película de la trilogía de Star Wars es su favorita. Evidentemente, sólo puede coger una, así que nos toca emplear radio buttons en lugar de casillas de verificación. Necesitamos entonces un botón por cada película (tres en total). ¿Cómo los agrupamos? Pues dando el mismo nombre de variable a cada botón. Es decir, el atributo **name siempre es el mismo para todo el grupo**. ¿Y qué hacemos con `id`? Bien, no puede haber dos valores o más de `id` repetidos, así que la `id` en cada botón ha de ser distinta. Es decir, los radio buttons son el único campo en el que `id` y `name` han de ser distintos. Disponemos además del atributo `checked` por si queremos poner algún valor por defecto. Nuestro código para este ejemplo sería así:

```

<p>
Peli preferida:<br />

<input type="radio" name="peli" id="peli_hope" value="hope"
checked="checked" />
<label for="peli_hope">A New Hope</label><br />

<input type="radio" name="peli" id="peli_empire"
value="empire" />
<label for="peli_empire">The Empire Strikes Back</label><br />

<input type="radio" name="peli" id="peli_jedi" value="jedi" />
<label for="peli_jedi">The Return of the Jedi</label><br />

```

---

<sup>3</sup>Yo soy cola, tú pegamento.

</p>

Usamos como nombre de variable (**name**) la palabra “*pele*”. Según la película que sea, cada botón tiene asignada una **id** distinta. Por ejemplo “*pelehope*” para *A New Hope*. El atributo **value** contiene el texto que tendrá la variable asignado en caso de que se seleccione ese botón. Por ejemplo, si el visitante selecciona la película del *Retorno del Jedi* en el mail que recibiríamos tendríamos una línea así:

```
pele=jedi
```

Quizás parezca un poco lioso, pero una vez que hayas hecho un par de formularios, todo irá bien.

## 9.8. Listas de selección

Las **listas de selección** tienen una función parecida a los radio buttons, en tanto que se nos presentan múltiples opciones agrupadas en las que escogemos una. La diferencia es que la lista aparece replegada y no ocupa apenas espacio en la web, así que son útiles cuando tenemos muchas opciones a elegir.

La etiqueta que las crea es `<select>` y tiene etiqueta de cierre. Entre ellas insertamos las opciones que tenemos con la etiqueta `<option>`. Pongamos el mismo ejemplo de antes, el de elegir la película preferida de *Star Wars*:

```
<p>
<label for="pele">Película preferida:</label><br />
<select name="pele" id="pele">
  <option value="hope">A New Hope</option>

  <option value="empire">
The Empire Strikes Back</option>

  <option value="jedi" selected="selected">
The Return of the Jedi</option>
</select>
</p>
```

Con `selected="selected"` indicamos cuál es la opción por defecto. En este caso, el Retorno del Jedi. Como ves, aquí no tenemos el **name** y **id** como con los botones de selección.

## 9.9. Botones de enviar y reestablecer

Ya hemos visto todos los campos de formulario que podemos crear, ahora sólo nos falta comentar dos **botones** especiales: el de **enviar** (*submit*) y el de

**reestablecer** (*reset*). Ambos se crean con `<input>`. El atributo `id` no tiene mucho sentido a no ser que usemos CSS para cambiar su aspecto de un modo en concreto y exclusivo. Asimismo, `name` no nos será muy útil si no empleamos algún tipo de script para tratar la información. Como nosotros lo mandamos por correo, no debemos preocuparnos. En `value` indicaremos el texto que aparezca en el botón.

El botón de **enviar** se encarga de mandar la información del formulario, según como lo hayamos especificado en `<form>` (en nuestro caso, manda los datos por e-mail). Simplemente especificamos `type="submit"`:

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

El botón de **reestablecer** borra el formulario y vuelve a poner los valores por defecto. Útil en formularios largos. Lo conseguimos con `type="reset"`:

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

Ni que decir tiene que hay que **diferenciar** bien cuál botón es cuál y no poner textos extraños como título de los botones. Fastidia mucho que se te borre todo el formulario por error, creyendo haberle dado al botón de enviar.

## Parte II

# CSS: La limoná

# Capítulo 10

## Introducción al CSS

En la parte de XHTML hemos comentado muchas veces que podemos cambiar el aspecto de una página web mediante CSS (*Cascading Style Sheets*). Ahora es el momento de aprender cómo hacerlo.

Me ha costado decidirme sobre cómo explicar las diferentes propiedades de CSS, pero creo que la mejor manera de aprender sin que se haga muy pesado es poniendo ejemplos de problemas comunes con los que todos nos topamos, junto con sus formas solución en CSS. Después de todo, esto no pretende ser una guía de referencia con todas las propiedades y todos sus posibles valores<sup>1</sup> ;)

### 10.1. ¿Dónde escribo la parte de CSS?

Hay dos opciones para insertar estilos en un documento XHTML. Lo más normal es hacerlo **en un archivo externo** (que se llama *hoja de estilos*) y enlazarlo desde nuestro documento XHTML. Esto tiene una ventaja muy grande y es que podemos tener muchas páginas enlazando a la misma hoja de estilos. Si más adelante queremos cambiar algo, sólo hemos de modificar un archivo (la hoja de estilos) y afectará a todas las páginas. Podemos enlazar una o más hojas poniendo esto dentro de la cabecera (head):

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

La otra opción es escribir la información referente a los estilos **en el mismo archivo XHTML**. Lo podemos hacer escribiendo entre las etiquetas `<style>` y `</style>`, que también deben ir en la cabecera.

Si hacemos las dos cosas a la vez, siempre tienen prioridad las reglas que estén dentro de `<style>`.

---

<sup>1</sup>Googleando un poco, podrás encontrar muchas. Y siempre te queda consultar las especificaciones en W3c.org



## 10.2. Vale, ¿pero cómo funciona?

En una hoja de estilos utilizamos reglas que consisten en elegir **selectores** a los que asignamos una serie de **propiedades**. Por ejemplo, si queremos que nuestra página web tenga el fondo blanco, las letras grises y use una fuente Arial de 11 píxeles de tamaño, escribimos:

```
body {  
  background-color:#fff;  
  color: #666;  
  font-family: Arial, sans-serif;  
  font-size: 11px;  
}
```

Si te lo estás preguntando, `#fff` significa el color blanco y `#666` un gris oscuro. Más adelante veremos cómo se representan los colores en CSS.

## 10.3. ¿Selectores?

Los **selectores** los utilizamos para elegir a qué elementos se aplican las propiedades que escribimos. Hay diferentes tipos de selectores, los más importantes son los que veremos ahora.

Si queremos elegir una **etiqueta**, simplemente escribimos su nombre. Por ejemplo, si queremos establecer propiedades para los enlaces:

```
a {  
  ...  
}
```

También podemos elegir un **elemento único** utilizando su atributo `id`. Para ello, empleamos la almohadilla:

```
#menu {  
  ...  
}
```

Otra cosa que podemos hacer es definir una **clase** y hacer que muchos elementos la utilicen, escribiendo un punto antes del nombre. Por ejemplo, si queremos destacar algo:

```
.importante {  
  ...  
}
```

Luego podríamos usar esa clase en los párrafos que queramos (o cualquier otro elemento), usando el atributo `class`. De este modo:

```
<p class="importante">Bla bla bla</p>
```

## 10.4. ¿Qué significa eso de Cascading?

*Cascading* significa cascada, y tiene que ver con la **herencia**. En CSS los elementos hijos heredan todas las propiedades de sus padres. Por ejemplo, si establecemos una regla para el elemento `table`, sus hijos (`td` entre otros) también tendrán esas mismas reglas.

Es por esto que si queremos establecer un tipo de fuente para todo el documento (por ejemplo) debemos indicarlo en el elemento `body`, porque todo nuestro contenido heredará de él.

## 10.5. ¿Puedo poner comentarios?

Para insertar un **comentario** dentro de una hoja de estilo o entre `<style>` y `</style>` lo hacemos así:

```
/* Esto es un comentario */
```

Los comentarios de XHTML, que van entre `<!--` y `-->` ya no nos sirven.

## 10.6. ¿Por qué fff significa blanco?

En CSS hay varias maneras de indicar un color. Podemos hacerlo en inglés, por ejemplo. En lugar de `#fff` escribimos `white` y lo solucionamos. El problema es que así es muy incómodo porque tenemos que aprendernos los nombres de cada color. Y puede que exista algún color que queramos que no tenga nombre. Así que normalmente se usa la **notación en hexadecimal**.

Los colores en nuestro monitor están formados por tres haces de luz: rojo, verde y azul. Se llama **sistema RGB** (Red Green Blue). Si cogemos un programa de dibujo, vemos que podemos coger un color indicando el valor de sus componentes rojo, verde y azul por separado. Este valor puede variar de 0 a 255 (se usan 8 bits para representar este rango). Por ejemplo, si queremos el amarillo puro, indicamos 255 para el rojo, 255 para el verde y 0 para el azul.

¿Qué tiene que ver esto con el sistema hexadecimal? Pues que este rango de valores (0..255) puede ser representado con dos dígitos hexadecimales, que van desde el 00 hasta el FF. La forma de indicar un color es con `#RRGGBB`, donde la almohadilla indica que se trata de un color hexadecimal y RR, GG y BB son los dígitos correspondientes al rojo, verde y azul. Entonces nuestro color amarillo sería `#ffff00`.

La mayoría de los colores que usaremos tendrán por cada componente los mismos dígitos. Por ejemplo `#ff0000` (rojo), `#ff00ff` (lila) o `#000000` (negro).

Así que los abreviamos a tres cifras. Entonces `#f00` es equivalente a `#ff0000`. Sin embargo, colores como `#a0a0a0` no pueden ser abreviados.

Para conseguir el color blanco debemos poner todas las componentes a 255, que sería `#ffffff` y abreviado nos quedaría `#fff`.

La mayoría de editores de código (X)HTML o de programas de dibujo nos mostrarán en la paleta el valor del color en hexadecimal. En cualquier caso, siempre podemos obtener el valor de sus componentes por separado y convertirlo a hexadecimal con una calculadora.

# Capítulo 11

## Trasteando por primera vez

Para empezar a trastear con CSS, conviene conocer algunas propiedades sencillas. ¡Empezamos!

### 11.1. El color de primer plano

La propiedad `color` hace referencia al *foreground color*, es decir, al **color** que está por encima del fondo. Hablando en plata, viene a ser el color del texto. Si queremos hacer que nuestra página tenga las letras de color gris oscuro, simplemente escribimos:

```
body { color:#666; }
```

Muy fácil, ¿no? Que pase el siguiente...

### 11.2. El fondo

Podemos modificar el **fondo** de un elemento con la propiedad `background`, que tiene la siguiente sintaxis:

```
background: [color] [image] [repeat] [attachment] [position]
```

El primer parámetro corresponde al **color** de fondo, los siguientes son relativos a la **imagen** de fondo.

- **image**: aquí indicamos qué imagen usaremos de fondo. Por ejemplo, `url("fondo.gif")`
- **repeat**: con esto establecemos si queremos que la imagen se repita o no, tanto horizontal como verticalmente. Con **repeat** se repite siempre en

ambos sentidos (valor por defecto), mientras que con `no-repeat` no se repite nunca. Con `repeat-x` se repite sólo horizontalmente y con `repeat-y` lo hace sólo en vertical.

- **attachment:** sirve para indicar si el fondo se queda fijo en el sitio o se desplaza con el scroll. Es un poco difícil de explicar, así que lo mejor es que lo probéis vosotros mismos: con `scroll` (valor por defecto) el fondo se desplaza y con `fixed` se queda siempre en el mismo sitio.
- **position:** indica la posición del fondo. Indicamos tanto la posición desde la izquierda como desde arriba (ya sea en píxeles o porcentaje). Los valores por defecto son `0% 0%` que sitúan al fondo en la esquina superior izquierda. Si lo quisieramos en las coordenadas 20,30 (tomando como origen esa esquina superior izquierda), escribiríamos `20px 30px`. Si queremos el fondo centrado, pues `50% 50%`.

Podemos omitir alguna propiedad si queremos. Además, podemos establecer los valores por separado, usando `background-color`, `background-repeat`, etc.

Veamos algunos ejemplos para poner fondo a nuestra página:

```
/* sólo color de fondo */
body { background-color: #fff; }

/* color de fondo e imagen en mosaico */
body { background: #fff url("fondo.gif");}

/* imagen fija, centrada y sin repetir, */
/* a modo de "marca de agua" */
body {
    background: #fff url("fondo.gif") no-repeat fixed 50% 50%;
}
```

Hemos explicado de forma bastante detallada la propiedad `background`, pero no te malacostumbres. Ahora iremos mucho más deprisa, y seguramente te toque buscar más información por tu cuenta :P

### 11.3. Fuente

Hay varias propiedades que nos permiten jugar con el aspecto del texto. Podemos englobar todas bajo `font`, pero primero veremos algunas sub-propiedades.

Lo primero que debemos considerar es que **no todos los ordenadores tienen las mismas fuentes instaladas**. Es decir, que mi maravillosa fuente llamada "Súper-Chachi-Way.ttf" no tiene por qué tenerla mi vecino. Es más, lo normal es que no la tenga<sup>1</sup>. ¿Entonces qué hacemos? Usar sólo fuentes

---

<sup>1</sup>En diseño web, como en los demás aspectos de la vida, la Ley de Murphy se suele cumplir.

“estándar”, que tengan la mayoría de ordenadores. Además, podemos especificar varias, de forma que si no se tiene la primera, se muestra la segunda, si no se tiene la segunda, pues la tercera, etc.

A lo que íbamos, la propiedad que sirve para cambiar la **fuentes** es `font-family`. Como acabamos de ver, podemos indicar varias fuentes, teniendo preferencia la primera. Si el nombre de una fuente tiene espacios en blanco, hay que ponerla entre comillas. Por ejemplo<sup>2</sup>:

```
font-family: "Trebuchet MS", Arial, sans-serif;
```

El **tamaño** de la fuente lo controlamos con `font-size`. Podemos indicar medidas en píxeles (px) o en puntos (pt)<sup>3</sup>.

El tema de la **negrita** se controla con `font-weight` asignándole `bold` o `normal`. La **cursiva** con `font-style`, poniendo `italic` o `normal`. Si queremos **versales**, pues `font-variant` con `small-caps` o `normal`. Como ves, todo muy intuitivo.

Resumiendo, todo junto quedaría tal que así:

```
background: [style] [variant] [weight] [size] [family]
```

Algunos ejemplillos:

```
/* fuente para todo el documento */
body {
    font: 11px Arial, Verdana, sans-serif;
    color: #666;
}

/* para el énfasis fuerte */
strong {
    font-variant: small-caps;
    color: #000;
}
```

## 11.4. Efectos en el texto

Además de cambios en la fuente, podemos alterar la forma en la que el texto es presentado. Aquí hay unas cuantas:

---

<sup>2</sup>Algunos opinan que la Trebuchet MS se convertirá pronto en la nueva Comic Sans MS. O sea, la típica fuente bonita que todo el mundo usa y que acaba cansando. Yo todavía no lo he hecho.

<sup>3</sup>Mentira cochina. Podemos (y muchos opinan que debemos) usar medidas relativas, pero el tema de la herencia es muy importante en estos casos y yo aquí no voy a explayarme. San Google bendito...

Para alinear el texto tenemos a `text-align`, que puede tomar los valores `right` (derecha), `left` (izquierda), `center` (centrado) o `justify` (justificado)<sup>4</sup>. Así que para centrar un párrafo:

```
p {text-align: center}
```

Si queremos **indentar** un párrafo, hacemos uso de la propiedad `text-indent`. Por ejemplo:

```
p { text-indent: 2em;}
```

¿Qué significa la unidad `em`? Es una unidad relativa. Lo más fácil es trabajar con píxeles o con porcentajes, pero en algunos casos conviene utilizar otras unidades. 1 `em` equivale al tamaño de la fuente. Así, si por herencia o por cualquier cosa, el texto se muestra en un tamaño de fuente mayor, la proporción de indentado siempre sería la misma. Si pusieramos `20px`, siempre sería esa, aunque el tamaño de fuente sea de `10px` o de `32px`, por ejemplo.

También podemos cambiar la **decoración** del texto con `text-decoration`. Puede tomar diversos valores, como `none` (sin ningún adorno), `underline` (subrayado)<sup>5</sup>, `overline` (subrayado superior) o `line-through` (tachado). Si queremos que nuestros links no tengan subrayado, por ejemplo, haríamos:

```
text-decoration: none;
```

Podemos también **transformar** las mayúsculas y minúsculas con `text-transform`. Si ponemos `lowercase` todo se mostrará en minúsculas; con `uppercase`, en mayúsculas; y con `capitalize` pondremos una letra capital al principio de cada palabra.

En cuanto al **espaciado**, para el espacio entre palabras usamos `word-spacing`; para el de las letras, `letter-spacing` y para el **interlineado**, `line-height`.

Un ejemplo con todo esto, para la etiqueta `strong`:

```
strong {
  color: #000;
  letter-spacing: 0.25em;
  text-transform: uppercase;
}
```

¿Demasiada información? Quizás, pero ten en cuenta que no hay que memorizar todo esto. Ni de coña. Lo importante es saber que **existe** una propiedad que hace tal cosa, no cómo se llama. Para eso están las guías de referencia rápida, y los editores de CSS.

---

<sup>4</sup>A mucha gente le gusta montar guerras sagradas acerca de la conveniencia o no de usar texto justificado. En caso de duda, pregunta a gente cómo le gusta más y asunto solucionado.

<sup>5</sup>¡Cuidado! En una web, si vemos algo subrayado, tendemos a pensar que es un enlace.

## Apéndice A

# Migración rápida a XHTML

Si ya usabas HTML 4 puedes aprender muy rápido XHTML, puesto que las bases son las mismas. Sólo hay que tener en cuenta que XHTML deja de lado todo aspecto estético y **se centra en el contenido y en la semántica**. En lugar de usar una etiqueta para dar un aspecto concreto, usamos etiquetas para hacer que las palabras signifiquen una cosa u otra. Ahora veremos una serie de “reglas” a seguir para pasarnos a XHTML. También es recomendable leer el capítulo 2, donde se explica la estructura de un documento XHTML. Evidentemente, no se muestran aquí todas las diferencias, pero sirve para hacernos una idea de qué es lo que nos espera<sup>1</sup>.

### A.1. Minúsculas y comillas, por favor

Antes era una práctica común escribir las etiquetas en mayúscula, para diferenciarlas del código. Por compatibilidad con XML, en XHTML todas las etiquetas deben ir en minúsculas.

Además, todos los atributos tiene que estar entre comillas dobles. Por ejemplo, si antes teníamos:

```
<IMG SRC=icono.gif ALT=icono>
```

Ahora escribimos:

```

```

---

<sup>1</sup>Diego Lafuente ha escrito artículos muy buenos en su blog [minid.net](http://www.minid.net) sobre el tema. Destaca el post titulado “13 pasos para dar el salto”. Podéis leerlo en [http://www.minid.net/archivos/categorias/xhtml/13\\_leyes\\_de\\_oro\\_para\\_dar\\_el\\_gran\\_paso.php](http://www.minid.net/archivos/categorias/xhtml/13_leyes_de_oro_para_dar_el_gran_paso.php)



## A.2. Todas las etiquetas se cierran

Con HTML podíamos crear párrafos con la etiqueta `<p>` sin necesidad de cerrarla con `</p>`. Lo mismo ocurría con `<li>`, por ejemplo. En XHTML **ninguna etiqueta puede quedar sin cerrar**.

Lo de ninguna es literal, y etiquetas que no tienen una de cierre, como `<img>`, `<br>`, etc deben ser cerradas. ¿Cómo lo hacemos? Insertando la barra `/`. Entonces la etiqueta para insertar un salto de línea es `<br />`. El espacio que hay entre el nombre y la barra es necesario para que navegadores antiguos reconozcan la etiqueta.

## A.3. FONT y ciertos atributos desaparecen

Hemos dicho que XHTML deja de lado la apariencia del documento, ya que eso es controlado por CSS. Entonces las etiquetas `<font>` y `<basefont>` carecen de sentido.

Además, atributos de algunas etiquetas que hacen referencia al **color** de las cosas o a imágenes de fondo, también desaparecen por este motivo. Son sustituidos por reglas CSS. Así que *sayônara* a `bgcolor` y amigos. Lo mismo para el atributo `align` usado en párrafos e imágenes.

## A.4. B y compañía también se van

Ciertas etiquetas de formato, como `<b>` (negrita), `<i>` (cursiva), etc ya no se usan porque hacen referencia exclusivamente a la **apariciencia** de las palabras. Si queremos dar énfasis, utilizamos `<em>` y para dar énfasis más fuerte, `<strong>`. Los navegadores suelen mostrarlas como cursiva y negrita, respectivamente (aunque cómo la representen es lo de menos, ya que lo podemos cambiar con CSS).

## A.5. Hay que usar alt y title

XHTML hace hincapié en la accesibilidad de un documento, y por eso debemos facilitar atributos de “apoyo” a algunas etiquetas. La etiqueta `<img>` dispone del atributo `alt` que se muestra cuando no se puede cargar la imagen y al pasar el cursor por encima. Hay que utilizarlo siempre.

Hay un atributo muy similar llamado `title` que se utiliza en la etiqueta `<a>`, entre otras, y sirve para mostrar una descripción del sitio al que nos dirigimos. Por ejemplo:

```
<a href="http://www.google.es" title="Buscador Google">Google</a>
```

## A.6. Cuidado al anidar etiquetas

XHTML es restrictivo en cuanto a la anidación de etiquetas se refiere. Básicamente hay dos tipos de elemento: los **block** y los **inline**. Los block son etiquetas como párrafos, títulos, listas, etc. Así a ojo los distinguimos porque siempre van solos e insertan saltos de línea. Los inline no interrumpen el flujo del texto. Son las etiquetas de formato, los enlaces y demás. **No podemos meter un elemento de bloque dentro de uno inline.**

Por ejemplo, si queremos insertar un link en nuestro título de la página, si lo hacemos de esta manera sería incorrecto:

```
<a href="..." title="..."><h1>texto</h1></a>
```

Hay que hacerlo así:

```
<h1><a href="..." title="...">texto</a></h1>
```

Además, hay ciertas etiquetas no admiten otras dentro. Si tienes dudas sobre qué elementos pueden ir dentro de otros, utiliza el **validador** de código del WWW Consortium<sup>2</sup>.

## A.7. No existen los frames

Aunque la especificación XHTML 1.0 Frameset permite el uso de frames (marcos) en una página, tanto la Transitional como la Strict los prohíben. Así que ya no podemos usar **ni frames ni inline frames**.

De todos modos, aunque se pudieran emplear, no deberíamos, ya que los frames son un atentado contra la usabilidad<sup>3</sup>.

## A.8. No se puede utilizar target

Antes la etiqueta `<a>` tenía un parámetro llamado **target** que permitía especificar en qué frame se cargaba el destino de un link. Como ya no hay frames, este atributo es innecesario.

A los amantes del `target="_blank"` para abrir webs en ventanas nueva les digo que si el usuario quiere abrir un ventanuco nuevo, lo hará con el botón derecho. Y para los que usamos navegación con pestañas (todos menos los usuarios de IE) es bastante molesto que se abran más ventanas del navegador. Gracias.

---

<sup>2</sup><http://www.w3c.org>

<sup>3</sup>Y vuelven loco al Google, de paso.

## A.9. Las tablas no se usan para maquetar

Aunque puedes crear un documento XHTML válido usando tablas para maquetar, va contra la filosofía de dejar XHTML sólo para contenido. Las tablas están hechas para ordenar información tabulada, no para diseñar layouts.

De todos modos, el paso para un layout tableless no suele ser fácil, así que creo que es más conveniente familiarizarse primero con el resto de XHTML y CSS, y dejar la maquetación sin tablas para más tarde<sup>4</sup>.

## A.10. Los & dan por saco

Si tienes URL's que contengan el **ampersand** (&), te llevarás la sorpresa de que el Validador del W3C se pelea con ellas. Esto es debido a las *entities*.

¿Qué demonios son? Pues la manera que tenemos de insertar caracteres especiales de forma "segura". Por ejemplo, el carácter á se escribiría `&acute;`. Como ves, las entities comienzan por un ampersand y terminan en punto y coma.

Por eso, cuando se leen URLs que contengan ampersands, hay confusión porque lo que sigue no es una entity... Así que lo que tenemos que hacer es sustituir el ampersand por su propia entity, que es `&amp;`.

Es decir, que esta URL:

```
http://www.alianza.net/main.php?nick=leia&show=10
```

Quedaría así:

```
http://www.alianza.net/main.php?nick=leia&amp;show=10
```

---

<sup>4</sup>“Tarde” no significa “nunca”.

## Apéndice B

# Licencia

Este documento está distribuido bajo la licencia de Creative Commons Attribution NonCommercial NoDerivs License<sup>1</sup>. Básicamente dice que puedes distribuir o mostrar el documento siempre que cumplas las siguientes condiciones:

- Des crédito y respetes la autoría (es decir, yo, Belén Albeza).
- No tengas ánimo de lucro ni saques provecho económico.
- No modifiques el documento ni lo utilices para hacer trabajos derivados.

Huy qué fácil.

---

<sup>1</sup>Para ver una copia de esta licencia, visita <http://creativecommons.org/licenses/by-nc-nd/2.0/> o manda una carta a Creative Commons, 559 Nathan Abbott Way, Stanford, California, USA.

## Apéndice C

# Sobre esta versión

Esta es una **versión preliminar** del documento definitivo. Faltan capítulos de XHTML y toda la parte de CSS. Además de imágenes que muestren la salida del código para hacer los ejemplos más útiles.

“¡Pues vaya mierda entonces!” Pues sí. Pero si lo he puesto on-line es por algo. Necesito que saques todos los errores que veas, mis meteduras de patas, me digas si hay algo que no está bien explicado, etc.

Lo dicho, que mi e-mail es benko@ladybenko.net Muchas gracias y atento a las nuevas versiones que se irán publicando en **demasiada Cafeína**<sup>1</sup> :)

**Actualización 9-9-2004:** Ya está casi acabada la sección de XHTML, a falta de las imágenes de ejemplo y alguna cosilla con el doctype y el IE 6. Se ha modificado el capítulo 2 y se han añadido el de Tablas y el de Formularios.

**Actualización 12-12-2004:** Añadido un apéndice para la migración a XHTML, así como los primeros capítulos de CSS.

---

<sup>1</sup><http://www.ladybenko.net/cafeina>