Índice

El gran libro de HTML5, CSS3 y Javascript

Juan Diego Gauchat

Página de créditos

El gran libro de HTML5, CSS3 y Javascript

Primera edición en libro electrónico: Enero de 2012

Gran Via de les Corts Catalanes, 594
08007 Barcelona (España)
www.marcombo.com

ISBN edición en formato electrónico: 978-84-267-1782-5

ISBN edición en papel: 978-84-267-1770-2

Introducción

HTML5 no es una nueva versión del antiguo lenguaje de etiquetas, ni siquiera una mejora de esta ya antigua tecnología, sino un nuevo concepto para la construcción de sitios web y aplicaciones en una era que combina dispositivos móviles, computación en la nube ytrabajos en red.

Todo comenzó mucho tiempo atrás con una simple versión de HTML propuesta para crear la estructura básica de páginas web, organizar su contenido y compartir información. El lenguaje y la web misma nacieron principalmente con la intención de comunicar información por medio de texto.

El limitado objetivo de HTML motivó a varias compañías a desarrollar nuevos lenguajes y programas para agregar características a la web nunca antes implementadas. Estos desarrollos iniciales crecieron hasta convertirse en populares y poderosos accesorios. Simples juegos y bromas animadas pronto se transformaron en sofisticadas aplicaciones, ofreciendo nuevas experiencias que cambiaron el concepto de la web para siempre.

De las opciones propuestas, Java y Flash fueron las más exitosas; ambas fueron masivamente adoptadas y ampliamente consideradas como el futuro de Internet. Sin embargo, tan pronto como el número de usuarios se incrementó e Internet pasó de ser una forma de conectar amantes de los ordenadores a un campo estratégico para los negocios y la interacción social, limitaciones presentes en estas dos tecnologías probaron ser una sentencia de muerte.

El mayor inconveniente de Java y Flash puede describirse como una falta de integración.Ambos fueron concebidos desde el principio como complementos (plug-ins), algo que se inserta dentro de una estructura pero que comparte con la misma solo espacio en la pantalla. No existía comunicación e integración alguna entre aplicaciones y documentos.

La falta de integración resultó ser crítica y preparó el camino para la evolución de un lenguaje que comparte espacio en el documento con HTML y no está afectado por las limitaciones de los *plug-ins*. Javascript, un lenguaje interpretado incluido en navegadores, claramente era la manera de mejorar la experiencia de los usuarios y proveer funcionalidad para la web. Sin embargo, después de algunos años de intentos fallidos para promoverlo y algunos malos usos, el mercado nunca lo adoptó plenamente y pronto su popularidad declinó. Los detractores tenían buenas razones para oponerse a su adopción. En ese momento, Javascript no era capaz de reemplazar la funcionalidad de Flash o Java. A pesar de ser evidente que ambos limitaban el alcance de las aplicaciones y aislaban el contenido web, populares funciones como la reproducción de video se estaban convirtiendo en una parte esencial de la web ysolo eran efectivamente ofrecidas a través de estas tecnologías.

Apesar del suceso inicial, el uso de Java comenzó a declinar. La naturaleza compleja del lenguaje, su evolución lenta y la falta de integración disminuyeron su importancia hasta el punto en el que hoy día no es más usado en aplicaciones web de importancia. Sin Java, el mercado volcó su atención a Flash. Pero el hecho de que Flash comparte las mismas características básicas que su competidor en la web lo hace también susceptible de correr el mismo destino.

Mientras esta competencia silenciosa se llevaba a cabo, el software para acceder a la web continuaba evolucionando. Junto con nuevas funciones y técnicas rápidas de acceso a la red, los navegadores también mejoraron gradualmente sus intérpretes Javascript. Más potencia trajo más oportunidades y este lenguaje estaba listo para aprovecharlas.

En cierto punto durante este proceso, se hizo evidente para algunos desarrolladores que ni Java o Flash podrían proveer las herramientas que ellos necesitaban para crear las aplicaciones demandadas por un número creciente de usuarios. Estos desarrolladores, impulsados por las mejoras otorgadas por los navegadores, comenzaron a aplicar Javascript en sus aplicaciones de un modo nunca visto. La innovación y los increíbles resultados obtenidos llamaron la atención de más programadores. Pronto lo que fue llamado la “Web 2.0” nació y la percepción de Javascript en la comunidad de programadores cambió radicalmente.

Javascript era claramente el lenguaje que permitía a los desarrolladores innovar y hacer cosas que nadie había podido hacer antes en la web. En los últimos años, programadores y diseñadores web alrededor del mundo surgieron con los más increíbles trucos para superar las limitaciones de esta tecnología y sus iniciales deficiencias en portabilidad. Gracias a estas nuevas implementaciones, Javascript, HTML y CSS se convirtieron pronto en la más perfecta combinación para la necesaria evolución de la web.

HTML5 es, de hecho, una mejora de esta combinación, el pegamento que une todo. HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas. Apartir de ahora, HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y Javascript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales.

Las barreras entre sitios webs y aplicaciones finalmente han desaparecido. Las tecnologías requeridas para el proceso de

integración están listas. El futuro de la web es prometedor y la evolución y combinación de estas tres tecnologías (HTML, CSS y Javascript) en una poderosa especificación está volviendo a Internet la plataforma líder de desarrollo. HTML5 indica claramente el camino.

> **IMPORTANTE:** En este momento no todos los navegadores soportan HTML5 y la mayoría de sus funciones se encuentran actualmente en estado de desarrollo. Recomendamos leer los capítulos y ejecutar los códigos con las últimas versiones de Google Chrome y Firefox. Google Chrome ya implementa muchas de las características de HTML5 y además es una buena plataforma para pruebas. Por otro lado, Firefox es uno de los mejores navegadores para desarrolladores ytambién provee total soporte para HTML5.
>
> Sea cual fuere el navegador elegido, siempre tenga en mente que un buen desarrollador instala y prueba sus códigos en cada programa disponible en el mercado. Ejecute los códigos provistos en este libro en cada uno de los navegadores disponibles.
>
> Para descargar las últimas versiones, visite los siguientes enlaces:
>
> · *www.google.com/chrome*
>
> · *www.apple.com/safari/download*
>
> · *www.mozilla.com*
>
> · *windows.microsoft.com*
>
> · *www.opera.com*

En la conclusión del libro exploramos diferentes alternativas para hacer sus sitios webs y aplicaciones accesibles desde viejos navegadores e incluso aquellos que aún no están preparados para HTML5.

Capítulo 1
Documentos HTML5

1.1 Componentes básicos

HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Nunca fue declarado oficialmente pero, incluso cuando algunas APIs (Interface de Programación de Aplicaciones) y la especificación de CSS3 por completo no son parte del mismo, HTML5 es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5. HTML está a cargo de la estructura, CSS presenta esa estructura y su contenido en la pantalla y Javascript hace el resto que (como veremos más adelante) es extremadamente significativo.

Más allá de esta integración, la estructura sigue siendo parte esencial de un documento. La misma provee los elementos necesarios para ubicar contenido estático o dinámico, y es también una plataforma básica para aplicaciones. Con la variedad de dispositivos para acceder a Internet y la diversidad de interfaces disponibles para interactuar con la web, un aspecto básico como la estructura se vuelve parte vital del documento. Ahora la estructura debe proveer forma, organización y flexibilidad, y debe ser tan fuerte como los fundamentos de un edificio.

Para trabajar y crear sitios webs y aplicaciones con HTML5, necesitamos saber primero cómo esa estructura es construida. Crear fundamentos fuertes nos ayudará más adelante a aplicar el resto de los componentes para aprovechar completamente estas nuevas tecnologías.

Por lo tanto, empecemos por lo básico, paso a paso. En este primer capítulo aprenderá cómo construir una plantilla para futuros proyectos usando los nuevos elementos HTML introducidos en HTML5.

> **Hágalo usted mismo:** Cree un archivo de texto vacío utilizando su editor de textos favorito para probar cada código presentado en este capítulo. Esto lo ayudará a recordar las nuevas etiquetas HTML y acostumbrarse a ellas.
>
> **Conceptos básicos:** Un documento HTML es un archivo de texto. Si usted no posee ningún programa para desarrollo web, puede simplemente utilizar el Bloc de Notas de Windows o cualquier otro editor de textos. El archivo debe ser grabado con la extensión **`.html`** y el nombre que desee (por ejemplo, **`micodigo.html`**).
>
> **IMPORTANTE:** Para acceder a información adicional y a los listados de ejemplo, visite nuestro sitio web *www.minkbooks.com*.

1.2 Estructura global

Los documentos HTML se encuentran estrictamente organizados. Cada parte del documento está diferenciada, declarada y determinada por etiquetas específicas. En esta parte del capítulo vamos a ver cómo construir la estructura global de un documento HTML ylos nuevos elementos semánticos incorporados en HTML5.

<!DOCTYPE>

En primer lugar necesitamos indicar el tipo de documento que estamos creando. Esto en HTML5 es extremadamente sencillo:

```
<!DOCTYPE html>
```

Listado 1-1. *Usando el elemento <doctype>.*

IMPORTANTE: Esta línea debe ser la primera línea del archivo, sin espacios o líneas que la precedan. De esta forma, el modo estándar del navegador es activado y las incorporaciones de HTML5 son interpretadas siempre que sea posible, o ignoradas en caso contrario.

Hágalo usted mismo: Puede comenzar a copiar el código en su archivo de texto y agregar los próximos a medida que los vamos estudiando.

<html>

Luego de declarar el tipo de documento, debemos comenzar a construir la estructura HTML. Como siempre, la estructura tipo árbol de este lenguaje tiene su raíz en el elemento **<html>**. Este elemento envolverá al resto del código:

```
<!DOCTYPE html>
<html lang="es">
</html>
```

Listado 1-2. *Usando el elemento <html>.*

El atributo **lang** en la etiqueta de apertura **<html>** es el único atributo que necesitamos especificar en HTML5. Este atributo define el idioma humano del contenido del documento que estamos creando, en este caso **es** por español.

Conceptos básicos: HTML usa un lenguaje de etiquetas para construir páginas web. Estas etiquetas HTML son palabras clave y atributos rodeados de los signos mayor y menor (por ejemplo, **<html lang="es">**). En este caso, **html** es la palabra clave y **lang** es el atributo con el valor **es**. La mayoría de las etiquetas HTML se utilizan en pares, una etiqueta de apertura y una de cierre, y el contenido se declara entre ellas. En nuestro ejemplo, **<html lang="es">** indica el comienzo del código HTML y **</html>** indica el final. Compare las etiquetas de apertura y cierre y verá que la de cierre se distingue por una barra invertida antes de la palabra clave (por ejemplo, **</html>**). El resto de nuestro código será insertado entre estas dos etiquetas: **<html> ... </html>**.

IMPORTANTE: HTML5 es extremadamente flexible en cuanto a la estructura y a los elementos utilizados para construirla. El elemento **<html>** puede ser incluido sin ningún atributo o incluso ignorado completamente. Con el propósito de preservar compatibilidad (y por algunas razones extras que no vale la pena mencionar aquí) le recomendamos que siga algunas reglas básicas. En este libro vamos a enseñarle cómo construir documentos HTML de acuerdo a lo que nosotros consideramos prácticas recomendadas.

Para encontrar otros lenguajes para el atributo **lang** puede visitar el siguiente enlace: *www.w3schools.com/tags/ref_language_codes.asp.*

<head>

Continuemos construyendo nuestra plantilla. El código HTML insertado entre las etiquetas **<html>** tiene que ser dividido entre dos secciones principales. Al igual que en versiones previas de HTML, la primera sección es la cabecera y la segunda el cuerpo. El siguiente paso, por lo tanto, será crear estas dos secciones en el código usando los elementos **<head>** y **<body>** ya

conocidos.

El elemento **<head>** va primero, por supuesto, y al igual que el resto de los elementos estructurales tiene una etiqueta de apertura y una de cierre:

```
<!DOCTYPE html>
<html lang="es">
<head>

</head>

</html>
```

Listado 1-3. *Usando el elemento <head>.*

La etiqueta no cambió desde versiones anteriores y su propósito sigue siendo exactamente el mismo. Dentro de las etiquetas **<head>** definiremos el título de nuestra página web, declararemos el set de caracteres correspondiente, proveeremos información general acerca del documento e incorporaremos los archivos externos con estilos, códigos Javascript o incluso imágenes necesarias para generar la página en la pantalla.

Excepto por el título y algunos íconos, el resto de la información incorporada en el documento entre estas etiquetas es invisible para el usuario.

<body>

La siguiente gran sección que es parte principal de la organización de un documento HTML es el cuerpo. El cuerpo representa la parte visible de todo documento y es especificado entre etiquetas **<body>**. Estas etiquetas tampoco han cambiado en relación con versiones previas de HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>

</head>
<body>

</body>
</html>
```

Listado 1-4. *Usando el elemento <body>.*

Conceptos básicos: Hasta el momento tenemos un código simple pero con una estructura compleja. Esto es porque el código HTML no está formado por un conjunto de instrucciones secuenciales. HTML es un lenguaje de etiquetas, un listado de elementos que usualmente se utilizan en pares y que pueden ser anidados (totalmente contenidos uno dentro del otro). En la primera línea del código del Listado 1-4 tenemos una etiqueta simple con la definición del tipo de documento e inmediatamente después la etiqueta de apertura **<html lang="es">**. Esta etiqueta y la de cierre **</html>** al final del listado están indicando el comienzo del código HTML y su final. Entre las etiquetas **<html>** insertamos otras etiquetas especificando dos importantes partes de la estructura básica: **<head>** para la cabecera y **<body>** para el cuerpo del documento. Estas dos etiquetas también se utilizan en pares. Más adelante en este capítulo veremos que más etiquetas son insertadas entre estas últimas conformando una estructura de árbol con **<html>** como su raíz.

<meta>

Es momento de construir la cabecera del documento. Algunos cambios e innovaciones fueron incorporados dentro de la cabecera, y uno de ellos es la etiqueta que define el juego de caracteres a utilizar para mostrar el documento. Ésta es una etiqueta **<meta>** que especifica cómo el texto será presentado en pantalla:

```
<!DOCTYPE html>
<html lang="es">
<head>
```

```
  <meta charset=”iso-8859-1”>

</head>
<body>

</body>
</html>
```

Listado 1-5. *Usando el elemento <meta>.*

La innovación de este elemento en HTML5, como en la mayoría de los casos, es solo simplificación. La nueva etiqueta **<meta>** para la definición del tipo de caracteres es más corta y simple. Por supuesto, podemos cambiar el tipo **iso-8859-1** por el necesario para nuestros documentos y agregar otras etiquetas **<meta>** como **description** o **keywords** para definir otros aspectos de la página web, como es mostrado en el siguiente ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name=”description” content=”Ejemplo de HTML5”>
  <meta name=”keywords” content=”HTML5, CSS3, Javascript”>

</head>
<body>

</body>
</html>
```

Listado 1-6. *Agregando más elementos <meta>.*

Conceptos básicos: Hay varios tipos de etiqueta **<meta>** que pueden ser incluidas para declarar información general sobre el documento, pero esta información no es mostrada en la ventana del navegador, es solo importante para motores de búsqueda y dispositivos que necesitan hacer una vista previa del documento u obtener un sumario de la información que contiene. Como comentamos anteriormente, aparte del título y algunos íconos, la mayoría de la información insertada entre las etiquetas **<head>** no es visible para los usuarios. En el código del Listado 1-6, el atributo **name** dentro de la etiqueta **<meta>** especifica su tipo y **content** declara su valor, pero ninguno de estos valores es mostrado en pantalla. Para aprender más sobre la etiqueta **<meta>**, visite nuestro sitio web y siga los enlaces proporcionados para este capítulo.

En HTML5 no es necesario cerrar etiquetas simples con una barra al final, pero recomendamos utilizarlas por razones de compatibilidad. El anterior código se podría escribir de la siguiente manera:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1" />
  <meta name="description" content="Ejemplo de HTML5" />
  <meta name="keywords" content="HTML5, CSS3, JavaScript" />

</head>
<body>

</body>
</html>
```

Listado 1-7. *Cierre de etiquetas simples.*

<title>

La etiqueta **<title>**, como siempre, simplemente especifica el título del documento, y no hay nada nuevo para comentar:

```
<!DOCTYPE html>
```

```
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>

</head>
<body>

</body>
</html>
```

Listado 1-8. *Usando la etiqueta <title>.*

Conceptos básicos: El texto entre las etiquetas **`<title>`** es el título del documento que estamos creando. Normalmente este texto es mostrado en la barra superior de la ventana del navegador.

<link>

Otro importante elemento que va dentro de la cabecera del documento es **`<link>`**. Este elemento es usado para incorporar estilos, códigos Javascript, imágenes o iconos desde archivos externos. Uno de los usos más comunes para **`<link>`** es la incorporación de archivos con estilos CSS:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel=”stylesheet” href=”misestilos.css”>
</head>
<body>

</body>
</html>
```

Listado 1-9. *Usando el elemento <link>.*

En HTML5 ya no se necesita especificar qué tipo de estilos estamos insertando, por lo que el atributo **`type`** fue eliminado. Solo necesitamos dos atributos para incorporar nuestro archivo de estilos: **`rel`** y **`href`**. El atributo **`rel`** significa “relación” y es acerca de la relación entre el documento y el archivo que estamos incorporando por medio de **`href`**. En este caso, el atributo **`rel`** tiene el valor **`stylesheet`** que le dice al navegador que el archivo **`misestilos.css`** es un archivo CSS con estilos requeridos para presentar la página en pantalla (en el próximo capítulo estudiaremos cómo utilizar estilos CSS).

Conceptos básicos: Un archivo de estilos es un grupo de reglas de formato que ayudarán a cambiar la apariencia de nuestra página web (por ejemplo, el tamaño y color del texto). Sin estas reglas, el texto y cualquier otro elemento HTML sería mostrado en pantalla utilizando los estilos estándar provistos por el navegador. Los estilos son reglas simples que normalmente requieren solo unas pocas líneas de código y pueden ser declarados en el mismo documento. Como veremos más adelante, no es estrictamente necesario obtener esta información de archivos externos pero es una práctica recomendada. Cargar las reglas CSS desde un documento externo (otro archivo) nos permitirá organizar el documento principal, incrementar la velocidad de carga y aprovechar las nuevas características de HTML5.

Con esta última inserción podemos considerar finalizado nuestro trabajo en la cabecera. Ahora es tiempo de trabajar en el cuerpo, donde la magia ocurre.

1.3 Estructura del cuerpo

La estructura del cuerpo (el código entre las etiquetas **`<body>`**) generará la parte visible del documento. Este es el código que producirá nuestra página web.

HTML siempre ofreció diferentes formas de construir y organizar la información dentro del cuerpo de un documento. Uno de los primeros elementos provistos para este propósito fue **<table>**. Las tablas permitían a los diseñadores acomodar datos, texto, imágenes y herramientas dentro de filas ycolumnas de celdas, incluso sin que hayan sido concebidas para este propósito.

En los primeros días de la web, las tablas fueron una revolución, un gran paso hacia adelante con respecto a la visualización de los documentos y la experiencia ofrecida a los usuarios. Más adelante, gradualmente, otros elementos reemplazaron su función, permitiendo lograr lo mismo con menos código, facilitando de este modo la creación, permitiendo portabilidad y ayudando al mantenimiento de los sitios web.

El elemento ** ** comenzó a dominar la escena. Con el surgimiento de webs más interactivas y la integración de HTML, CSS y Javascript, el uso de ** ** se volvió una práctica común. Pero este elemento, así como **<table>**, no provee demasiada información acerca de las parte del cuerpo que está representando. Desde imágenes a menús, textos, enlaces, códigos, formularios, cualquier cosa puede ir entre las etiquetas de apertura y cierre de un elemento ** **. En otras palabras, la palabra clave **div** solo especifica una división en el cuerpo, como la celda de una tabla, pero no ofrece indicio alguno sobre qué clase de división es, cuál es su propósito o qué contiene.

Para los usuarios estas claves o indicios no son importantes, pero para los navegadores la correcta interpretación de qué hay dentro del documento que se está procesando puede ser crucial en muchos casos. Luego de la revolución de los dispositivos móviles y el surgimiento de diferentes formas en que la gente accede a la web, la identificación de cada parte del documento es una tarea que se ha vuelto más relevante que nunca.

Considerando todo lo expuesto, HTML5 incorpora nuevos elementos que ayudan a identificar cada sección del documento y organizar el cuerpo del mismo. En HTML5 las secciones más importantes son diferenciadas y la estructura principal ya no depende más de los elementos ** ** o **<table>**.

Cómo usamos estos nuevos elementos depende de nosotros, pero las palabras clave otorgadas a cada uno de ellos nos dan ayudan a entender sus funciones. Normalmente una página o aplicación web está dividida entre varias áreas visuales para mejorar la experiencia del usuario y facilitar la interactividad. Las palabras claves que representan cada nuevo elemento de HTML5 están íntimamente relacionadas con estas áreas, como veremos pronto.

Organización

La Figura 1-1 representa un diseño común encontrado en la mayoría de los sitios webs estos días.Apesar del hecho de que cada diseñador crea sus propios diseños, en general podremos identificar las siguientes secciones en cada sitio web estudiado:

Figura 1-1. *Representación visual de un clásico diseño web.*

En la parte superior, descripto como **Cabecera**, se encuentra el espacio donde usualmente se ubica el logo, título, subtítulos y una corta descripción del sitio web o la página.

Inmediatamente debajo, podemos ver la **Barra de Navegación** en la cual casi todos los desarrolladores ofrecen un menú o lista de enlaces con el propósito de facilitar la navegación a través del sitio. Los usuarios son guiados desde esta barra hacia las diferentes páginas o documentos, normalmente pertenecientes al mismo sitio web.

El contenido más relevante de una página web se encuentra, en casi todo diseño, ubicado en su centro. Esta sección presenta información y enlaces valiosos. La mayoría de las veces es dividida en varias filas y columnas. En el ejemplo de la Figura 1-1 se utilizaron solo dos columnas: **Información Principal** y **Barra Lateral**, pero esta sección es extremadamente flexible y normalmente diseñadores la adaptan acorde a sus necesidades insertando más columnas, dividiendo cada columna entre bloques más pequeños o generando diferentes distribuciones y combinaciones. El contenido presentado en esta parte del diseño es usualmente de alta prioridad. En el diseño de ejemplo, **Información Principal** podría contener una lista de artículos, descripción de productos, entradas de un blog o cualquier otra información importante, y la **Barra Lateral** podría mostrar una lista de enlaces apuntando hacia cada uno se esos ítems. En un blog, por ejemplo, esta última columna ofrecerá una lista de enlaces apuntando a cada entrada del blog, información acerca del autor, etc…

En la base de un diseño web clásico siempre nos encontramos con una barra más que aquí llamamos **Institucional**. La nombramos de esta manera porque esta es el área en donde normalmente se muestra información acerca del sitio web, el autor o la empresa, además de algunos enlaces con respecto a reglas, términos y condiciones y toda información adicional que el desarrollador considere importante compartir. La barra **Institucional** es un complemento de la **Cabecera** y es parte de lo que se considera estos días la estructura esencial de una página web, como podemos apreciar en el siguiente ejemplo:

Figura 1-2. *Representación visual de un clásico diseño para blogs.*

La Figura 1-2 es una representación de un blog normal. En este ejemplo se puede claramente identificar cada parte del diseño considerado anteriormente.

1. **Cabecera**
2. **Barra de Navegación**

3. Sección de **Información Principal**
4. **Barra Lateral**
5. El pie o la barra **Institucional**

Esta simple representación de un blog nos puede ayudar a entender que cada sección definida en un sitio web tiene un propósito. A veces este propósito no es claro pero en esencia se encuentra siempre allí, ayudándonos a reconocer cualquiera de las secciones descriptas anteriormente en todo diseño.

HTML5 considera esta estructura básica y provee nuevos elementos para diferenciar y declarar cada una de sus partes. A partir de ahora podemos decir al navegador para qué es cada sección:

Figura 1-3. *Representación visual de un diseño utilizando elementos HTML5.*

La Figura 1-3 muestra el típico diseño presentado anteriormente, pero esta vez con los correspondientes elementos HTML5 para cada sección (incluyendo etiquetas de apertura ycierre).

<header>

Uno de los nuevos elementos incorporados en HTML5 es **<header>**. El elemento **<header>** no debe ser confundido con **<head>** usado antes para construir la cabecera del documento. Del mismo modo que **<head>**, la intención de **<header>** es proveer información introductoria (títulos, subtítulos, logos), pero difiere con respecto a **<head>** en su alcance. Mientras que el elemento **<head>** tiene el propósito de proveer información acerca de todo el documento, **<header>** es usado solo para el cuerpo o secciones específicas dentro del cuerpo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <h1>Este es el título principal del sitio web</h1>
  </header>

</body>
</html>
```

Listado 1-10. *Usando el elemento <header>.*

En el Listado 1-10, definimos el título de la página web utilizando el elemento **<header>**. Recuerde que esta cabecera no es la misma que la utilizada previamente para definir el título del documento. La inserción del elemento **<header>** representa el comienzo del cuerpo y por lo tanto de la parte visible del documento. De ahora en más será posible ver los resultados de nuestro código en la ventana del navegador.

Hágalo usted mismo: Si siguió las instrucciones desde el comienzo de este capítulo ya debería contar con un archivo de texto creado con todos los códigos estudiados hasta el momento y listo para ser probado. Si no es así, todo lo que debe hacer es copiar el código en el Listado 1-10 dentro de un archivo de texto vacío utilizando cualquier editor de texto (como el Bloc de Notas de Windows, por ejemplo) y grabar el archivo con el nombre de su agrado y la extensión **.html**. Para ver el código en funcionamiento, abra el archivo en un navegador compatible con HTML5 (puede hacerlo con un doble clic sobre el archivo en su explorador de archivos).

Conceptos básicos: Entre las etiquetas **<header>** en el Listado 1-10 hay un elemento que probablemente no conoce. El elemento **<h1>** es un viejo elemento HTML usado para definir títulos. El número indica la importancia del título. El elemento **<h1>** es el más importante y **<h6>** el de menor importancia, por lo tanto **<h1>** será utilizado para mostrar el título principal y los demás para subtítulos o subtítulos internos. Más adelante veremos cómo estos elementos trabajan en HTML5.

<nav>

Siguiendo con nuestro ejemplo, la siguiente sección es la **Barra de Navegación**. Esta barra es generada en HTML5 con el elemento **<nav>**:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <h1>Este es el título principal del sitio web</h1>
  </header>
  <nav>
    <ul>
      <li>principal</li>
      <li>fotos</li>
      <li>videos</li>
      <li>contacto</li>
    </ul>
  </nav>

</body>
</html>
```

Listado 1-11. *Usando el elemento <nav>.*

Como se puede apreciar en el Listado 1-11, el elemento **<nav>** se encuentra dentro de las etiquetas **<body>** pero es ubicado después de la etiqueta de cierre de la cabecera (**</header>**), no dentro de las etiquetas **<header>**. Esto es porque **<nav>** no es parte de la cabecera sino una nueva sección.

Anteriormente dijimos que la estructura y el orden que elegimos para colocar los elementos HTML5 dependen de nosotros. Esto significa que HTML5 es versátil y solo nos otorga los parámetros y elementos básicos con los que trabajar, pero cómo usarlos será exclusivamente decisión nuestra. Un ejemplo de esta versatilidad es que el elemento **<nav>** podría ser insertado dentro del elemento **<header>** o en cualquier otra parte del cuerpo. Sin embargo, siempre se debe considerar que estas etiquetas fueron creadas para brindar información a los navegadores y ayudar a cada nuevo programa y dispositivo en el mercado a identificar las partes más relevantes del documento. Para conservar nuestro código portable y comprensible, recomendamos como buena práctica seguir lo que marcan los estándares y mantener todo tan claro como sea posible. El elemento **<nav>** fue creado para ofrecer ayuda para la navegación, como en menús principales o grandes bloques de enlaces, y debería ser utilizado de esa manera.

> **Conceptos básicos:** En el ejemplo del Listado 1-11 generamos las opciones del menú para nuestra página web. Entre las etiquetas **<nav>** hay dos elementos que son utilizados para crear una lista. El propósito del elemento **<ul>** es definir la lista. Anidado entre las etiquetas **<ul>** encontramos varias etiquetas **<li>** con diferentes textos representando las opciones del menú. Las etiquetas **<li>**, como probablemente ya se ha dado cuenta, son usadas para definir cada ítem de la lista. El propósito de este libro no es enseñarle conceptos básicos sobre HTML, si necesita más información acerca de elementos regulares de este lenguaje visite nuestro sitio web y siga los enlaces correspondientes a este capítulo.

<section>

Siguiendo nuestro diseño estándar nos encontramos con las columnas que en la Figura 1-1 llamamos **Información Principal** y **Barra Lateral**. Como explicamos anteriormente, la columna **Información Principal** contiene la información más relevante del documento y puede ser encontrada en diferentes formas (por ejemplo, dividida en varios bloques o columnas). Debido a que el propósito de estas columnas es más general, el elemento en HTML5 que especifica estas secciones se llama simplemente **<section>**:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <h1>Este es el título principal del sitio web</h1>
  </header>
  <nav>
    <ul>
      <li>principal</li>
      <li>fotos</li>
      <li>videos</li>
      <li>contacto</li>
    </ul>
  </nav>
  <section>

  </section>

</body>
</html>
```

Listado 1-12. *Usando el elemento <section>.*

Al igual que la **Barra de Navegación**, la columna **Información Principal** es una sección aparte. Por este motivo, la sección para **Información Principal** va debajo de la etiqueta de cierre **`</nav>`**.

> **Hágalo usted mismo:** Compare el último código en el Listado 1-12 con el diseño de la Figura 1-3 para comprender cómo las etiquetas son ubicadas en el código y qué sección cada una de ellas genera en la representación visual de la página web.
>
> **IMPORTANTE:** Las etiquetas que representan cada sección del documento están localizadas en el código en forma de lista, unas sobre otras, pero en el sitio web algunas de estas secciones se ubicarán lado a lado (las columnas **Información Principal** y **Barra Lateral** son un claro ejemplo). En HTML5, la responsabilidad por la representación de los elementos en la pantalla fue delegada a CSS. El diseño será logrado asignando estilos CSS a cada elemento HTML. Estudiaremos CSS en el próximo capítulo.

<aside>

En un típico diseño web (Figura 1-1) la columna llamada **Barra Lateral** se ubica al lado de la columna **Información Principal**. Esta es una columna o sección que normalmente contiene datos relacionados con la información principal pero que no son relevantes o igual de importantes.

En el diseño de un blog, por ejemplo, la **Barra Lateral** contendrá una lista de enlaces. En el ejemplo de la Figura 1-2, los enlaces apuntan a cada una de las entradas del blog y ofrecen información adicional sobre el autor (número 4). La información dentro de esta barra está relacionada con la información principal pero no es relevante por sí misma. Siguiendo el mismo ejemplo podemos decir que las entradas del blog son relevantes pero los enlaces y las pequeñas reseñas sobre esas entradas son solo una ayuda para la navegación pero no lo que al lector realmente le interesa.

En HTML5 podemos diferenciar esta clase secundaria de información utilizando el elemento **`<aside>`**:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
```

```
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <h1>Este es el título principal del sitio web</h1>
  </header>
  <nav>
    <ul>
      <li>principal</li>
      <li>fotos</li>
      <li>videos</li>
      <li>contacto</li>
    </ul>
  </nav>
  <section>

   </section>
   <aside>
     <blockquote>Mensaje número uno</blockquote>
     <blockquote>Mensaje número dos</blockquote>
   </aside>

</body>
</html>
```

Listado 1-13. *Usando el elemento <aside>.*

El elemento **<aside>** podría estar ubicado del lado derecho o izquierdo de nuestra página de ejemplo, la etiqueta no tiene una posición predefinida. El elemento **<aside>** solo describe la información que contiene, no el lugar dentro de la estructura. Este elemento puede estar ubicado en cualquier parte del diseño y ser usado siempre y cuando su contenido no sea considerado como el contenido principal del documento. Por ejemplo, podemos usar **<aside>** dentro del elemento **<section>** o incluso insertado entre la información relevante, como en el caso de una cita.

<footer>

Para finalizar la construcción de la plantilla o estructura elemental de nuestro documento HTML5, solo necesitamos un elemento más. Ya contamos con la cabecera del cuerpo, secciones con ayuda para la navegación, información importante y hasta una barra lateral con datos adicionales, por lo tanto lo único que nos queda por hacer es cerrar nuestro diseño para otorgarle un final al cuerpo del documento. HTML5 provee un elemento específico para este propósito llamado **<footer>**:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <h1>Este es el título principal del sitio web</h1>
  </header>
  <nav>
    <ul>
      <li>principal</li>
      <li>fotos</li>
      <li>videos</li>
```

```
      <li>contacto</li>
    </ul>
  </nav>
  <section>

  </section>
  <aside>
    <blockquote>Mensaje número uno</blockquote>
    <blockquote>Mensaje número dos</blockquote>
  </aside>
  <footer>
    Derechos Reservados &copy; 2010-2011
  </footer>

</body>
</html>
```

Listado 1-14. *Usando el elemento <footer>.*

En el típico diseño de una página web (Figura 1-1) la sección llamada **Institucional** será definida por etiquetas **<footer>**. Esto es debido a que la barra representa el final (o pie) del documento y esta parte de la página web es normalmente usada para compartir información general sobre el autor o la organización detrás del proyecto.

Generalmente, el elemento **<footer>** representará el final del cuerpo de nuestro documento y tendrá el propósito descripto anteriormente. Sin embargo, **<footer>** puede ser usado múltiples veces dentro del cuerpo para representar también el final de diferentes secciones (del mismo modo que la etiqueta **<header>**). Estudiaremos esta última característica más adelante.

1.4 Dentro del cuerpo

El cuerpo de nuestro documento está listo. La estructura básica de nuestro sitio web fue finalizada, pero aún tenemos que trabajar en el contenido. Los elementos HTML5 estudiados hasta el momento nos ayudan a identificar cada sección del diseño y asignar un propósito intrínseco a cada una de ellas, pero lo que es realmente importante para nuestro sitio web se encuentra en el interior de estas secciones.

La mayoría de los elementos ya estudiados fueron creados para construir una estructura para el documento HTML que pueda ser identificada y reconocida por los navegadores y nuevos dispositivos. Aprendimos acerca de la etiqueta **<body>** usada para declarar el cuerpo o parte visible del documento, la etiqueta **<header>** con la que agrupamos información importante para el cuerpo, la etiqueta **<nav>** que provee ayuda para la navegación del sitio web, la etiqueta **<section>** necesaria para contener la información más relevante, y también **<aside>** y **<footer>** para ofrecer información adicional de cada sección y del documento mismo. Pero ninguno de estos elementos declara algo acerca del contenido. Todos tienen un específico propósito estructural.

Más profundo nos introducimos dentro del documento más cerca nos encontramos de la definición del contenido. Esta información estará compuesta por diferentes elementos visuales como títulos, textos, imágenes, videos y aplicaciones interactivas, entre otros. Necesitamos poder diferenciar estos elementos y establecer una relación entre ellos dentro de la estructura.

<article>

El diseño considerado anteriormente (Figura 1-1) es el más común y representa una estructura esencial para los sitios web estos días, pero es además ejemplo de cómo el contenido clave es mostrado en pantalla. Del mismo modo que los blogs están divididos en entradas, sitios web normalmente presentan información relevante dividida en partes que comparten similares características. El elemento **<article>** nos permite identificar cada una de estas partes:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
```

```
</head>
<body>
  <header>
    <h1>Este es el título principal del sitio web</h1>
  </header>
  <nav>
    <ul>
      <li>principal</li>
      <li>fotos</li>
      <li>videos</li>
      <li>contacto</li>
    </ul>
  </nav>
  <section>
    <article>
      Este es el texto de mi primer mensaje
    </article>
    <article>
      Este es el texto de mi segundo mensaje
    </article>
  </section>
  <aside>
    <blockquote>Mensaje número uno</blockquote>
    <blockquote>Mensaje número dos</blockquote>
  </aside>
  <footer>
    Derechos Reservados &copy; 2010-2011
  </footer>
</body>
</html>
```

Listado 1-15. *Usando el elemento <article>.*

Como puede observarse en el código del Listado 1-15, las etiquetas **<article>** se encuentran ubicadas dentro del elemento **<section>**. Las etiquetas **<article>** en nuestro ejemplo pertenecen a esta sección, son sus hijos, del mismo modo que cada elemento dentro de las etiquetas **<body>** es hijo del cuerpo. Y al igual que cada elemento hijo del cuerpo, las etiquetas **<article>** son ubicadas una sobre otra, como es mostrado en la Figura 1-4.

Conceptos básicos: Como dijimos anteriormente, la estructura de un documento HTML puede ser descripta como un árbol, con el elemento **<html>** como su raíz. Otra forma de describir la relación entre elementos es nombrarlos como padres, hijos y hermanos, de acuerdo a la posición que ocupan dentro de esa misma estructura. Por ejemplo, en un típico documento HTML el elemento **<body>** es hijo del elemento **<html>** y hermano del elemento **<head>**. Ambos, **<body>** y **<head>**, tienen al elemento **<html>** como su padre.

Figura 1-4. *Representación visual de las etiquetas <article> que fueron incluidas para contener información relevante de la página web.*

El elemento **<article>** no está limitado por su nombre (no se limita, por ejemplo, a artículos de noticias). Este elemento fue creado con la intención de contener unidades independientes de contenido, por lo que puede incluir mensajes de foros, artículos de una revista digital, entradas de blog, comentarios de usuarios, etc… Lo que hace es agrupar porciones de información que están relacionadas entre sí independientemente de su naturaleza.

Como una parte independiente del documento, el contenido de cada elemento **<article>** tendrá su propia estructura. Para definir esta estructura, podemos aprovechar la versatilidad de los elementos **<header>** y **<footer>** estudiados anteriormente. Estos elementos son portables y pueden ser usados no solo para definir los límites del cuerpo sino también en cualquier sección de nuestro documento:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <h1>Este es el título principal del sitio web</h1>
  </header>
  <nav>
    <ul>
      <li>principal</li>
      <li>fotos</li>
      <li>videos</li>
      <li>contacto</li>
    </ul>
  </nav>
  <section>
    <article>
      <header>
        <h1>Título del mensaje uno</h1>
```

```
      </header>
      Este es el texto de mi primer mensaje
      <footer>
        <p>comentarios (0)</p>
      </footer>
    </article>
    <article>
      <header>
        <h1>Titulo del mensaje dos</h1>
      </header>
      Este es el texto de mi segundo mensaje
      <footer>
        <p>comentarios (0)</p>
      </footer>
    </article>
  </section>
  <aside>
    <blockquote>Mensaje número uno</blockquote>
    <blockquote>Mensaje número dos</blockquote>
  </aside>
  <footer>
    Derechos Reservados &copy; 2010-2011
  </footer>
</body>
</html>
```

Listado 1-16. *Construyendo la estructura de <article>.*

Los dos mensajes insertados en el código del Listado 1-16 fueron construidos con el elemento **<article>** y tienen una estructura específica. En la parte superior de esta estructura incluimos las etiquetas **<header>** conteniendo el título definido con el elemento **<h1>**, debajo se encuentra el contenido mismo del mensaje y sobre el final, luego del texto, vienen las etiquetas **<footer>** especificando la cantidad de comentarios recibidos.

<hgroup>

Dentro de cada elemento **<header>**, en la parte superior del cuerpo o al comienzo de cada **<article>**, incorporamos elementos **<h1>** para declarar un título. Básicamente, las etiquetas **<h1>** son todo lo que necesitamos para crear una línea de cabecera para cada parte del documento, pero es normal que necesitemos también agregar subtítulos o más información que especifique de qué se trata la página web o una sección en particular. De hecho, el elemento **<header>** fue creado para contener también otros elementos como tablas de contenido, formularios de búsqueda o textos cortos ylogos.

Para construir este tipo de cabeceras, podemos aprovechar el resto de las etiquetas H, como **<h1>, <h2>**, **<h3>**, **<h4>**, **<h5>** y **<h6>**, pero siempre considerando que por propósitos de procesamiento interno, y para evitar generar múltiples secciones durante la interpretación del documento por parte del navegador, estas etiquetas deben ser agrupadas juntas. Por esta razón, HTML5 provee el elemento **<hgroup>**:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <h1>Este es el título principal del sitio web</h1>
  </header>
  <nav>
    <ul>
```

```
      <li>principal</li>
      <li>fotos</li>
      <li>videos</li>
      <li>contacto</li>
    </ul>
  </nav>
  <section>
    <article>
      <header>
        <hgroup>
          <h1>Título del mensaje uno</h1>
          <h2>Subtítulo del mensaje uno</h2>
        </hgroup>
        <p>publicado 10-12-2011</p>
      </header>
      Este es el texto de mi primer mensaje
      <footer>
        <p>comentarios (0)</p>
      </footer>
    </article>
    <article>
      <header>
        <hgroup>
          <h1>Título del mensaje dos</h1>
          <h2>Subtítulo del mensaje dos</h2>
        </hgroup>
        <p>publicado 15-12-2011</p>
      </header>
      Este es el texto de mi segundo mensaje
      <footer>
        <p>comentarios (0)</p>
      </footer>
    </article>
  </section>
  <aside>
    <blockquote>Mensaje número uno</blockquote>
    <blockquote>Mensaje número dos</blockquote>
  </aside>
  <footer>
    Derechos Reservados &copy; 2010-2011
  </footer>
</body>
</html>
```

Listado 1-17. *Usando el elemento <hgroup>.*

Las etiquetas H deben conservar su jerarquía, lo que significa que debemos primero declarar la etiqueta **<h1>**, luego usar **<h2>** para subtítulos y así sucesivamente. Sin embargo, a diferencia de anteriores versiones de HTML, HTML5 nos deja reusar las etiquetas H yconstruir esta jerarquía una y otra vez en cada sección del documento. En el ejemplo del Listado 1-17, agregamos un subtítulo y datos adicionales a cada mensaje. Los títulos y subtítulos fueron agrupados juntos utilizando **<hgroup>**, recreando de este modo la jerarquía **<h1>** y **<h2>** en cada elemento **<article>**.

> **IMPORTANTE:** El elemento **<hgroup>** es necesario cuando tenemos un título y subtítulo o más etiquetas H juntas en la misma cabecera. Este elemento puede contener solo etiquetas H y esta fue la razón por la que en nuestro ejemplo dejamos los datos adicionales afuera. Si solo dispone de una etiqueta **<h1>** o la etiqueta **<h1>** junto con datos adicionales, no tiene que agrupar estos elementos juntos. Por ejemplo, en la cabecera del cuerpo (**<header>**) no usamos este elemento porque solo tenemos una etiqueta H en su interior. Siempre recuerde que **<hgroup>** fue creado solo con la intención de agrupar etiquetas H, exactamente como su nombre lo indica.

Navegadores y programas que ejecutan y presentan en la pantalla sitios webs leen el código HTML y crean su propia estructura interna para interpretar y procesar cada elemento. Esta estructura interna está dividida en secciones que no tienen nada que ver con las divisiones en el diseño o el elemento **<section>**. Estas son secciones conceptuales generadas durante la interpretación del código. El elemento **<header>** no crea una de estas secciones por sí mismo, lo que significa que los elementos dentro de **<header>** representarán diferentes niveles e internamente pueden generar diferentes secciones. El elemento **<hgroup>** fue creado con el propósito de agrupar las etiquetas H y evitar interpretaciones incorrectas por parte de los

navegadores.

Conceptos básicos: lo que llamamos “información adicional” dentro de la cabecera en nuestra descripción previa es conocido como Metadata. Metadata es un conjunto de datos que describen y proveen información acerca de otro grupo de datos. En nuestro ejemplo, Metadata es la fecha en la cual cada mensaje fue publicado.

<figure> y <figcaption>

La etiqueta **<figure>** fue creada para ayudarnos a ser aún más específicos a la hora de declarar el contenido del documento.Antes de que este elemento sea introducido, no podíamos identificar el contenido que era parte de la información pero a la vez independiente, como ilustraciones, fotos, videos, etc… Normalmente estos elementos son parte del contenido relevante pero pueden ser extraídos o movidos a otra parte sin afectar o interrumpir el flujo del documento. Cuando nos encontramos con esta clase de información, las etiquetas **<figure>** pueden ser usadas para identificarla:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <h1>Este es el título principal del sitio web</h1>
  </header>
  <nav>
    <ul>
      <li>principal</li>
      <li>fotos</li>
      <li>videos</li>
      <li>contacto</li>
    </ul>
  </nav>
  <section>
    <article>
      <header>
        <hgroup>
          <h1>Título del mensaje uno</h1>
          <h2>Subtítulo del mensaje uno</h2>
        </hgroup>
        <p>publicado 10-12-2011</p>
      </header>
      Este es el texto de mi primer mensaje
      <figure>
        <img src=”http://minkbooks.com/content/myimage.jpg”>
        <figcaption>
           Esta es la imagen del primer mensaje
         </figcaption>
      </figure>
      <footer>
        <p>comentarios (0)</p>
      </footer>
    </article>
    <article>
      <header>
        <hgroup>
          <h1>Título del mensaje dos</h1>
          <h2>Subtítulo del mensaje dos</h2>
        </hgroup>
        <p>publicado 15-12-2011</p>
```

```
      </header>
      Este es el texto de mi segundo mensaje
      <footer>
        <p>comentarios (0)</p>
      </footer>
    </article>
  </section>
  <aside>
    <blockquote>Mensaje número uno</blockquote>
    <blockquote>Mensaje número dos</blockquote>
  </aside>
  <footer>
    Derechos Reservados &copy; 2010-2011
  </footer>
</body>
</html>
```

Listado 1-18. *Usando los elementos <figure> y <figcaption>.*

En el Listado 1-18, en el primer mensaje, luego del texto insertamos una imagen (**`<img src="http://minkbooks.com/content/myimage.jpg">`**). Esta es una práctica común, a menudo el texto es enriquecido con imágenes o videos. Las etiquetas **`<figure>`** nos permiten envolver estos complementos visuales y diferenciarlos así de la información más relevante.

También en el Listado 1-18 se puede observar un elemento extra dentro de **`<figure>`**. Normalmente, unidades de información como imágenes o videos son descriptas con un corto texto debajo. HTML5 provee un elemento para ubicar e identificar estos títulos descriptivos. Las etiquetas **`<figcaption>`** encierran el texto relacionado con **`<figure>`** y establecen una relación entre ambos elementos ysu contenido.

1.5 Nuevos y viejos elementos

HTML5 fue desarrollado con la intención de simplificar, especificar y organizar el código. Para lograr este propósito, nuevas etiquetas y atributos fueron agregados y HTML fue completamente integrado a CSS y Javascript. Estas incorporaciones y mejoras de versiones previas están relacionadas no solo con nuevos elementos sino también con cómo usamos los ya existentes.

<mark>

La etiqueta **`<mark>`** fue agregada para resaltar parte de un texto que originalmente no era considerado importante pero ahora es relevante acorde con las acciones del usuario. El ejemplo que más se ajusta a este caso es un resultado de búsqueda. El elemento **`<mark>`** resaltará la parte del texto que concuerda con el texto buscado:

```
<span>Mi <mark>coche</mark> es rojo</span>
```

Listado 1-19. *Uso del elemento <mark> para resaltar la palabra "coche".*

Si un usuario realiza una búsqueda de la palabra "coche", por ejemplo, los resultados podrían ser mostrados con el código del Listado 1-19. La frase del ejemplo representa los resultados de la búsqueda y las etiquetas **`<mark>`** en el medio encierran lo que era el texto buscado (la palabra "coche"). En algunos navegadores, esta palabra será resaltada con un fondo amarillo por defecto, pero siempre podemos sobrescribir estos estilos con los nuestros utilizando CSS, como veremos en próximos capítulos.

En el pasado, normalmente obteníamos similares resultados usando el elemento **`<b>`**. El agregado de **`<mark>`** tiene el objetivo de cambiar el significado y otorgar un nuevo propósito para éstos y otros elementos relacionados:

- **`<em>`** es para indicar énfasis (reemplazando la etiqueta **`<i>`** que utilizábamos anteriormente).
- **`<strong>`** es para indicar importancia.
- **`<mark>`** es para resaltar texto que es relevante de acuerdo con las circunstancias.
- **`<b>`** debería ser usado solo cuando no hay otro elemento más apropiado para la situación.

<small>

La nueva especificidad de HTML es también evidente en elementos como **<small>**. Previamente este elemento era utilizado con la intención de presentar cualquier texto con letra pequeña. La palabra clave referenciaba el tamaño del texto, independientemente de su significado. En HTML5, el nuevo propósito de **<small>** es presentar la llamada letra pequeña, como impresiones legales, descargos, etc…

```
<small>Derechos Reservados &copy; 2011 MinkBooks</small>
```

Listado 1-20. *Inclusión de información legal con el elemento <small>.*

<cite>

Otro elemento que ha cambiado su naturaleza para volverse más específico es **<cite>**. Ahora las etiquetas **<cite>** encierran el título de un trabajo, como un libro, una película, una canción, etc…

```
<span>Amo la película <cite>Tentaciones</cite></span>
```

Listado 1-21. *Citando una película con el elemento <cite>.*

<address>

El elemento **<address>** es un viejo elemento convertido en un elemento estructural. No necesitamos usarlo previamente para construir nuestra plantilla, sin embargo podría ubicarse perfectamente en algunas situaciones en las que debemos presentar información de contacto relacionada con el contenido del elemento **<article>** o el cuerpo completo.

Este elemento debería ser incluido dentro de **<footer>,** como en el siguiente ejemplo:

```
<article>
  <header>
    <h1>Título del mensaje </h1>
  </header>
  Este es el texto del mensaje
  <footer>
    <address>
      <a href="http://www.jdgauchat.com">JD Gauchat</a>
    </address>
  </footer>
</article>
```

Listado 1-22. *Agregando información de contacto a un <article>.*

<time>

En cada **<article>** de nuestra última plantilla (Listado 1-18), incluimos la fecha indicando cuándo el mensaje fue publicado. Para esto usamos un simple elemento **<p>** dentro de la cabecera (**<header>**) del mensaje, pero existe un elemento en HTML5 específico para este propósito. El elemento **<time>** nos permite declarar un texto comprensible para humanos y navegadores que representa fecha y hora:

```
<article>
  <header>
    <h1>Título del mensaje dos</h1>
    <time datetime="2011-10-12" pubdate>publicado 12-10-2011</time>
  </header>
  Este es el texto del mensaje
```

```
</article>
```

Listado 1-23. *Fecha y hora usando el elemento <time>.*

En el Listado 1-23, el elemento **<p>** usado en ejemplos previos fue reemplazado por el nuevo elemento **<time>** para mostrar la fecha en la que el mensaje fue publicado. El atributo **datetime** tiene el valor que representa la fecha comprensible para el navegador (timestamp). El formato de este valor deberá seguir un patrón similar al del siguiente ejemplo: **2011-10-12T12:10:45**. También incluimos el atributo **pubdate**, el cual solo es agregado para indicar que el valor del atributo **datetime** representa la fecha de publicación.

1.6 Referencia rápida

En la especificación HTML5, HTML está a cargo de la estructura del documento y provee un grupo completo de nuevos elementos para este propósito. La especificación también incluye algunos elementos con la única tarea de proveer estilos. Esta es una lista de los que consideramos más relevantes:

IMPORTANTE: Para una completa referencia de los elementos HTML incluidos en la especificación, visite nuestro sitio web ysiga los enlaces correspondientes a este capítulo.

<header> Este elemento presenta información introductoria y puede ser aplicado en diferentes secciones del documento. Tiene el propósito de contener la cabecera de una sección pero también puede ser utilizado para agrupar índices, formularios de búsqueda, logos, etc…

<nav> Este elemento indica una sección de enlaces con propósitos de navegación, como menús o índices. No todos los enlaces dentro de una página web tienen que estar dentro de un elemento **<nav>**, solo aquellos que forman partes de bloques de navegación.

<section> Este elemento representa una sección general del documento. Es usualmente utilizado para construir varios bloques de contenido (por ejemplo, columnas) con el propósito de ordenar el contenido que comparte una característica específica, como capítulos o páginas de un libro, grupo de noticias, artículos, etc…

<aside> Este elemento representa contenido que está relacionado con el contenido principal pero no es parte del mismo. Ejemplos pueden ser citas, información en barras laterales, publicidad, etc…

<footer> Este elemento representa información adicional sobre su elemento padre. Por ejemplo, un elemento **<footer>** insertado al final del cuerpo proveerá información adicional sobre el cuerpo del documento, como el pie normal de una página web. Puede ser usado no solo para el cuerpo sino también para diferentes secciones dentro del cuerpo, otorgando información adicional sobre estas secciones específicas.

<article> Este elemento representa una porción independiente de información relevante (por ejemplo, cada artículo de un periódico o cada entrada de un blog). El elemento **<article>** puede ser anidado y usado para mostrar una lista dentro de otra lista de ítems relacionados, como comentarios de usuarios en entradas de blogs, por ejemplo.

<hgroup> Este elemento es usado para agrupar elementos H cuando la cabecera tiene múltiples niveles (por ejemplo, una cabecera con título ysubtítulo).

<figure> Este elemento representa una porción independiente de contenido (por ejemplo, imágenes, diagramas o videos) que son referenciadas desde el contenido principal. Esta es información que puede ser removida sin afectar el fluido del resto del contenido.

<figcaption> Este elemento es utilizado para mostrar una leyenda o pequeño texto relacionado con el contenido de un elemento **<figure>**, como la descripción de una imagen.

<mark> Este elemento resalta un texto que tiene relevancia en una situación en particular o que ha sido mostrado en respuesta de la actividad del usuario.

<small> Este elemento representa contenido al margen, como letra pequeña (por ejemplo, descargos, restricciones legales, declaración de derechos, etc…).

<cite> Este elemento es usado para mostrar el título de un trabajo (libro, película, poema, etc…).

<address> Este elemento encierra información de contacto para un elemento **<article>** o el documento completo. Es recomendable que sea insertado dentro de un elemento **<footer>**.

<time> Este elemento se utiliza para mostrar fecha y hora en formatos comprensibles por los usuarios y el navegador. El valor para los usuarios es ubicado entre las etiquetas mientras que el específico para programas y navegadores es incluido como el valor del atributo **datetime**. Un segundo atributo optativo llamado **pubdate** es usado para indicar que el valor de **datetime** es la fecha de publicación.

Capítulo 2
Estilos CSS y modelos de caja

2.1 CSS y HTML

Como aclaramos anteriormente, la nueva especificación de HTML (HTML5) no describe solo los nuevos elementos HTML o el lenguaje mismo. La web demanda diseño yfuncionalidad, no solo organización estructural o definición de secciones. En este nuevo paradigma, HTML se presenta junto con CSS yJavascript como un único instrumento integrado.

La función de cada tecnología ya ha sido explicada en capítulos previos, así como los nuevos elementos HTML responsables de la estructura del documento. Ahora es momento de analizar CSS, su relevancia dentro de esta unión estratégica ysu influencia sobre la presentación de documentos HTML.

Oficialmente CSS nada tiene que ver con HTML5. CSS no es parte de la especificación y nunca lo fue. Este lenguaje es, de hecho, un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML. Al comienzo, atributos dentro de las etiquetas HTML proveían estilos esenciales para cada elemento, pero a medida que el lenguaje evolucionó, la escritura de códigos se volvió más compleja y HTML por sí mismo no pudo más satisfacer las demandas de diseñadores. En consecuencia, CSS pronto fue adoptado como la forma de separar la estructura de la presentación. Desde entonces, CSS ha crecido y ganado importancia, pero siempre desarrollado en paralelo, enfocado en las necesidades de los diseñadores y apartado del proceso de evolución de HTML.

La versión 3 de CSS sigue el mismo camino, pero esta vez con un mayor compromiso. La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño. Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta es la razón por la que cada vez que mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas.

En este momento las nuevas características incorporadas en CSS3 están siendo implementadas e incluidas junto al resto de la especificación en navegadores compatibles con HTML5. En este capítulo, vamos a estudiar conceptos básicos de CSS y las nuevas técnicas de CSS3 ya disponibles para presentación y estructuración. También aprenderemos cómo utilizar los nuevos selectores y pseudo clases que hacen más fácil la selección e identificación de elementos HTML.

> **Conceptos básicos:** CSS es un lenguaje que trabaja junto con HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, etc…
>
> **IMPORTANTE:** En este momento las nuevas incorporaciones de CSS3 están siendo implementadas en las últimas versiones de los navegadores más populares, pero algunas de ellas se encuentran aún en estado experimental. Por esta razón, estos nuevos estilos deberán ser precedidos por prefijos tales como **`-moz-`** o **`-webkit-`** para ser efectivamente interpretados. Analizaremos este importante asunto más adelante.

2.2 Estilos y estructura

A pesar de que cada navegador garantiza estilos por defecto para cada uno de los elementos HTML, estos estilos no necesariamente satisfacen los requerimientos de cada diseñador. Normalmente se encuentran muy distanciados de lo que queremos para nuestros sitios webs. Diseñadores y desarrolladores a menudo deben aplicar sus propios estilos para obtener la organización y el efecto visual que realmente desean.

> **IMPORTANTE:** En esta parte del capítulo vamos a revisar estilos CSS y explicar algunas técnicas básicas para definir la estructura de un documento. Si usted ya se encuentra familiarizado con estos conceptos, siéntase libre de obviar las partes que ya conoce.

Elementos block

Con respecto a la estructura, básicamente cada navegador ordena los elementos por defecto de acuerdo a su tipo: *block* (bloque) o *inline* (en línea). Esta clasificación está asociada con la forma en que los elementos son mostrados en pantalla.

> Elementos block son posicionados uno sobre otro hacia abajo en la página.
>
> Elementos inline son posicionados lado a lado, uno al lado del otro en la misma línea, sin ningún salto de línea a menos

que ya no haya más espacio horizontal para ubicarlos.

Casi todos los elementos estructurales en nuestros documentos serán tratados por los navegadores como elementos *block* por defecto. Esto significa que cada elemento HTML que representa una parte de la organización visual (por ejemplo, **`<section>`**, **`<nav>`**, **`<header>`**, **`<footer>`**, **`<div>`**) será posicionado debajo del anterior.

En el Capítulo 1 creamos un documento HTML con la intención de reproducir un sitio web tradicional. El diseño incluyó barras horizontales y dos columnas en el medio. Debido a la forma en que los navegadores muestran estos elementos por defecto, el resultado en la pantalla está muy lejos de nuestras expectativas. Tan pronto como el archivo HTML con el código del Listado 1-18, Capítulo 1, es abierto en el navegador, la posición errónea en la pantalla de las dos columnas definidas por los elementos **`<section>`** y **`<aside>`** es claramente visible. Una columna está debajo de la otra en lugar de estar a su lado, como correspondería. Cada bloque (*block*) es mostrado por defecto tan ancho como sea posible, tan alto como la información que contiene y uno sobre otro, como se muestra en la Figura 2-1.

Figura 2-1. *Representación visual de una página web mostrada con estilos por defecto.*

Modelos de caja

Para aprender cómo podemos crear nuestra propia organización de los elementos en pantalla, debemos primero entender cómo los navegadores procesan el código HTML. Los navegadores consideran cada elemento HTML como una caja. Una página web es en realidad un grupo de cajas ordenadas siguiendo ciertas reglas. Estas reglas son establecidas por estilos provistos por los navegadores o por los diseñadores usando CSS.

CSS tiene un set predeterminado de propiedades destinados a sobrescribir los estilos provistos por navegadores y obtener la organización deseada. Estas propiedades no son específicas, tienen que ser combinadas para formar reglas que luego serán usadas para agrupar cajas y obtener la correcta disposición en pantalla. La combinación de estas reglas es normalmente llamada modelo o sistema de disposición. Todas estas reglas aplicadas juntas constituyen lo que se llama un modelo de caja.

Existe solo un modelo de caja que es considerado estándar estos días, y muchos otros que aún se encuentran en estado experimental. El modelo válido y ampliamente adoptado es el llamado Modelo de Caja Tradicional, el cual ha sido usado desde la primera versión de CSS.

Aunque este modelo ha probado ser efectivo, algunos modelos experimentales intentan superar sus deficiencias, pero la falta de consenso sobre el reemplazo más adecuado aún mantiene a este viejo modelo en vigencia y la mayoría de los sitios webs programados en HTML5 lo continúan utilizando.

2.3 Conceptos básicos sobre estilos

Antes de comenzar a insertar reglas CSS en nuestro archivo de estilos y aplicar un modelo de caja, debemos revisar los conceptos básicos sobre estilos CSS que van a ser utilizados en el resto del libro.

Aplicar estilos a los elementos HTML cambia la forma en que estos son presentados en pantalla. Como explicamos anteriormente, los navegadores proveen estilos por defecto que en la mayoría de los casos no son suficientes para satisfacer las necesidades de los diseñadores. Para cambiar esto, podemos sobrescribir estos estilos con los nuestros usando diferentes técnicas.

> **Conceptos básicos:** En este libro encontrará solo una introducción breve a los estilos CSS. Solo mencionamos las técnicas y propiedades que necesita conocer para entender los temas y códigos estudiados en próximos capítulos. Si considera que no tiene la suficiente experiencia en CSS y necesita mayor información visite nuestro sitio web y siga los enlaces correspondientes a este capítulo.
>
> **Hágalo usted mismo:** Dentro de un archivo de texto vacío, copie cada código HTML estudiado en los siguientes listados y abra el archivo en su navegador para comprobar su funcionamiento. Tenga en cuenta que el archivo debe tener la extensión **`.html`** para ser abierto y procesado correctamente.

Estilos en línea

Una de las técnicas más simples para incorporar estilos CSS a un documento HTML es la de asignar los estilos dentro de las etiquetas por medio del atributo **`style`**.

El Listado 2-1 muestra un documento HTML simple que contiene el elemento **`<p>`** modificado por el atributo **`style`** con el valor **`font-size: 20px`**. Este estilo cambia el tamaño por defecto del texto dentro del elemento **`<p>`** a un nuevo tamaño de 20 pixeles.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este es el título del documento</title>
</head>
<body>
  <p style="font-size: 20px">Mi texto</p>
</body>
</html>
```

Listado 2-1. *Estilos CSS dentro de etiquetas HTML.*

Usar la técnica demostrada anteriormente es una buena manera de probar estilos y obtener una vista rápida de sus efectos, pero no es recomendado para aplicar estilos a todo el documento. La razón es simple: cuando usamos esta técnica, debemos escribir y repetir cada estilo en cada uno de los elementos que queremos modificar, incrementando el tamaño del documento a proporciones inaceptables y haciéndolo imposible de mantener y actualizar. Solo imagine lo que ocurriría si decide que en lugar de 20 pixeles el tamaño de cada uno de los elementos **`<p>`** debería ser de 24 pixeles. Tendría que modificar cada estilo en cada etiqueta **`<p>`** en el documento completo.

Estilos embebidos

Una mejor alternativa es insertar los estilos en la cabecera del documento y luego usar referencias para afectar los elementos HTML correspondientes:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <style>
    p { font-size: 20px }
  </style>
```

```
</head>
<body>
  <p>Mi texto</p>
</body>
</html>
```

Listado 2-2. *Estilos listados en la cabecera del documento.*

El elemento **<style>** (mostrado en el Listado 2-2) permite a los desarrolladores agrupar estilos CSS dentro del documento. En versiones previas de HTML era necesario especificar qué tipo de estilos serían insertados. En HTML5 los estilos por defecto son CSS, por lo tanto no necesitamos agregar ningún atributo en la etiqueta de apertura **<style>**.

El código resaltado del Listado 2-2 tiene la misma función que la línea de código del Listado 2-1, pero en el Listado 2-2 no tuvimos que escribir el estilo dentro de cada etiqueta **<p>** porque todos los elementos **<p>** ya fueron afectados. Con este método, reducimos nuestro código y asignamos los estilos que queremos a elementos específicos utilizando referencias. Veremos más sobre referencias en este capítulo.

Archivos externos

Declarar los estilos en la cabecera del documento ahorra espacio y vuelve al código más consistente y actualizable, pero nos requiere hacer una copia de cada grupo de estilos en todos los documentos de nuestro sitio web. La solución es mover todos los estilos a un archivo externo y luego utilizar el elemento **<link>** para insertar este archivo dentro de cada documento que los necesite. Este método nos permite cambiar los estilos por completo simplemente incluyendo un archivo diferente. También nos permite modificar o adaptar nuestros documentos a cada circunstancia o dispositivo, como veremos al final del libro.

En el Capítulo 1, estudiamos la etiqueta **<link>** y cómo utilizarla para insertar archivos con estilos CSS en nuestros documentos. Utilizando la línea **<link rel="stylesheet" href="misestilos.css">** le decimos al navegador que cargue el archivo **misestilos.css** porque contiene todos los estilos necesitados para presentar el documento en pantalla. Esta práctica fue ampliamente adoptada por diseñadores que ya están trabajando con HTML5. La etiqueta **<link>** referenciando el archivo CSS será insertada en cada uno de los documentos que requieren de esos estilos:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <p>Mi texto</p>
</body>
</html>
```

Listado 2-3. *Aplicando estilos CSS desde un archivo externo.*

Hágalo usted mismo: De ahora en adelante agregaremos estilos CSS al archivo llamado **misestilos.css**. Debe crear este archivo en el mismo directorio (carpeta) donde se encuentra el archivo HTML y copiar los estilos CSS en su interior para comprobar cómo trabajan.

Conceptos básicos: Los archivos CSS son archivos de texto comunes. Al igual que los archivos HTML, puede crearlos utilizando cualquier editor de texto como el Bloc de Notas de Windows, por ejemplo.

Referencias

Almacenar todos nuestros estilos en un archivo externo e insertar este archivo dentro de cada documento que lo necesite es muy conveniente, sin embargo no podremos hacerlo sin buenos mecanismos que nos ayuden a establecer una específica relación entre estos estilos ylos elementos del documento que van a ser afectados.

Cuando hablábamos sobre cómo incluir estilos en el documento, mostramos una de las técnicas utilizadas a menudo en CSS para referenciar elementos HTML. En el Listado 2-2, el estilo para cambiar el tamaño de la letra referenciaba cada elemento **<p>** usando la palabra clave **p**. De esta manera el estilo insertado entre las etiquetas **<style>** referenciaba cada etiqueta **<p>** del documento y asignaba ese estilo particular a cada una de ellas.

Existen varios métodos para seleccionar cuáles elementos HTML serán afectados por las reglas CSS: ·

referencia por la palabra clave del elemento

· referencia por el atributo **id**

· referencia por el atributo **class**

Más tarde veremos que CSS3 es bastante flexible a este respecto e incorpora nuevas y más específicas técnicas para referenciar elementos, pero por ahora aplicaremos solo estas tres.

Referenciando con palabra clave

Al declarar las reglas CSS utilizando la palabra clave del elemento afectamos cada elemento de la misma clase en el documento. Por ejemplo, la siguiente regla cambiará los estilos de todos los elementos **<p>**:

```
p { font-size: 20px }
```

Listado 2-4. *Referenciando por palabra clave.*

Esta es la técnica presentada previamente en el Listado 2-2. Utilizando la palabra clave **p** al frente de la regla le estamos diciendo al navegador que esta regla debe ser aplicada a cada elemento **<p>** encontrado en el documento HTML. Todos los textos envueltos en etiquetas **<p>** tendrán el tamaño de 20 pixeles.

Por supuesto, lo mismo funcionará para cualquier otro elemento HTML. Si especificamos la palabra clave **span** en lugar de **p**, por ejemplo, cada texto entre etiquetas **** tendrá un tamaño de 20 pixeles:

```
span { font-size: 20px }
```

Listado 2-5. *Referenciando por otra palabra clave.*

¿Pero qué ocurre si solo necesitamos referenciar una etiqueta específica? ¿Debemos usar nuevamente el atributo **style** dentro de esta etiqueta? La respuesta es no. Como aprendimos anteriormente, el método de **Estilos en Línea** (usando el atributo **style** dentro de etiquetas HTML) es una técnica en desuso y debería ser evitada. Para seleccionar un elemento HTML específico desde las reglas de nuestro archivo CSS, podemos usar dos atributos diferentes: **id** y **class**.

Referenciando con el atributo id

El atributo **id** es como un nombre que identifica al elemento. Esto significa que el valor de este atributo no puede ser duplicado. Este nombre debe ser único en todo el documento.

Para referenciar un elemento en particular usando el atributo **id** desde nuestro archivo CSS la regla debe ser declarada con el símbolo **#** al frente del valor que usamos para identificar el elemento:

```
#texto1 { font-size: 20px }
```

Listado 2-6. *Referenciando a través del valor del atributo id.*

La regla en el Listado 2-6 será aplicada al elemento HTML identificado con el atributo **id="texto1"**. Ahora nuestro código HTML lucirá de esta manera:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
```

```
<body>
  <p id=”texto1”>Mi texto</p>
</body>
</html>
```

Listado 2-7. *Identificando el elemento <p> a través de su atributo id.*

El resultado de este procedimiento es que cada vez que hacemos una referencia usando el identificador **texto1** en nuestro archivo CSS, el elemento con ese valor de **id** será modificado, pero el resto de los elementos **<p>**, o cualquier otro elemento en el mismo documento, no serán afectados.

Esta es una forma extremadamente específica de referenciar un elemento y es normalmente utilizada para elementos más generales, como etiquetas estructurales. El atributo **id** y su especificidad es de hecho más apropiado para referencias en Javascript, como veremos en próximos capítulos.

Referenciando con el atributo class

La mayoría del tiempo, en lugar de utilizar el atributo **id** para propósitos de estilos es mejor utilizar **class**. Este atributo es más flexible y puede ser asignado a cada elemento HTML en el documento que comparte un diseño similar:

```
.texto1 { font-size: 20px }
```

Listado 2-8. *Referenciando por el valor del atributo class.*

Para trabajar con el atributo **class,** debemos declarar la regla CSS con un punto antes del nombre. La ventaja de este método es que insertar el atributo **class** con el valor **texto1** será suficiente para asignar estos estilos a cualquier elemento que queramos:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <p class=”texto1”>Mi texto</p>
  <p class=”texto1”>Mi texto</p>
  <p>Mi texto</p>
</body>
</html>
```

Listado 2-9. *Asignando estilos a varios elementos a través del atributo class.*

Los elementos **<p>** en las primeras dos líneas dentro del cuerpo del código en el Listado 2-9 tienen el atributo **class** con el valor **texto1**. Como dijimos previamente, la misma regla puede ser aplicada a diferentes elementos en el mismo documento. Por lo tanto, estos dos primeros elementos comparten la misma regla y ambos serán afectados por el estilo del Listado 2-8. El último elemento **<p>** conserva los estilos por defecto otorgados por el navegador.

La razón por la que debemos utilizar un punto delante del nombre de la regla es que es posible construir referencias más complejas. Por ejemplo, se puede utilizar el mismo valor para el atributo **class** en diferentes elementos pero asignar diferentes estilos para cada tipo:

```
p.texto1 { font-size: 20px }
```

Listado 2-10. *Referenciando solo elementos <p> a través del valor del atributo class.*

En el Listado 2-10 creamos una regla que referencia la clase llamada **texto1** pero solo para los elementos de tipo **<p>**. Si cualquier otro elemento tiene el mismo valor en su atributo **class** no será modificado por esta regla en particular.

Referenciando con cualquier atributo

Aunque los métodos de referencia estudiados anteriormente cubren un variado espectro de situaciones, a veces no son suficientes para encontrar el elemento exacto. La última versión de CSS ha incorporado nuevas formas de referenciar elementos HTML. Uno de ellas es el **Selector de Atributo**. Ahora podemos referenciar un elemento no solo por los atributos **`id`** y **`class`** sino también a través de cualquier otro atributo:

```
p[name] { font-size: 20px }
```

Listado 2-11. *Referenciando solo elementos <p> que tienen el atributo name.*

La regla en el Listado 2-11 cambia solo elementos **`<p>`** que tienen un atributo llamado **`name`**. Para imitar lo que hicimos previamente con los atributos **`id`** y **`class`**, podemos también especificar el valor del atributo:

```
p[name=”mitexto”] { font-size: 20px }
```

Listado 2-12. *Referenciando elementos <p> que tienen un atributo name con el valor mitexto.*

CSS3 permite combinar “**=**” con otros para hacer una selección más específica:

```
p[name^=”mi”] { font-size: 20px }
p[name$=”mi”] { font-size: 20px }
p[name*=”mi”] { font-size: 20px }
```

Listado 2-13. *Nuevos selectores en CSS3.*

Si usted conoce **Expresiones Regulares** desde otros lenguajes como Javascript o PHP, podrá reconocer los selectores utilizados en el Listado 2-13. En CSS3 estos selectores producen similares resultados:

- La regla con el selector **`^=`** será asignada a todo elemento **`<p>`** que contiene un atributo **`name`** con un valor comenzado en “**`mi`**” (por ejemplo, “**`mitexto`**”, “**`micasa`**”).
- La regla con el selector **`$=`** será asignada a todo elemento **`<p>`** que contiene un atributo **`name`** con un valor finalizado en “**`mi`**” (por ejemplo “**`textomi`**”, “**`casami`**”).
- La regla con el selector **`*=`** será asignada a todo elemento **`<p>`** que contiene un atributo **`name`** con un valor que incluye el texto “**`mi`**” (en este caso, el texto podría también encontrarse en el medio, como en “**`textomicasa`**”).

En estos ejemplos usamos el elemento **`<p>`**, el atributo **`name`**, y una cadena de texto al azar como “**`mi`**”, pero la misma técnica puede ser utilizada con cualquier atributo y valor que necesitemos. Solo tiene que escribir los corchetes e insertar entre ellos el nombre del atributo y el valor que necesita para referenciar el elemento HTML correcto.

Referenciando con pseudo clases

CSS3 también incorpora nuevas pseudo clases que hacen la selección aún más específica.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <div id="wrapper">
    <p class="mitexto1">Mi texto1</p>
    <p class="mitexto2">Mi texto2</p>
    <p class="mitexto3">Mi texto3</p>
    <p class="mitexto4">Mi texto4</p>
  </div>
```

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

Listado 2-14. *Plantilla para probar pseudo clases.*

Miremos por un momento el nuevo código HTML del Listado 2-14. Contiene cuatro elementos **<p>** que, considerando la estructura HTML, son hermanos entre sí e hijos del mismo elemento ** **.

Usando pseudo clases podemos aprovechar esta organización y referenciar un elemento específico sin importar cuánto conocemos sobre sus atributos y el valor de los mismos:

```
p:nth-child(2){
  background: #999999;
}
```

Listado 2-15. *Pseudo clase nth-child().*

La pseudo clase es agregada usando dos puntos luego de la referencia y antes del su nombre. En la regla del Listado 2- 15 referenciamos solo elementos **<p>**. Esta regla puede incluir otras referencias. Por ejemplo, podríamos escribirla como **.miclase:nth-child(2)** para referenciar todo elemento que es hijo de otro elemento y tiene el valor de su atributo **class** igual a **miclase**. La pseudo clase puede ser aplicada a cualquier tipo de referencia estudiada previamente.

La pseudo clase **nth-child()** nos permite encontrar un hijo específico. Como ya explicamos, el documento HTML del Listado 2-14 tiene cuatro elementos **<p>** que son hermanos. Esto significa que todos ellos tienen el mismo padre que es el elemento ** **. Lo que esta pseudo clase está realmente indicando es algo como: “el hijo en la posición…” por lo que el número entre paréntesis será el número de la posición del hijo, o índice. La regla del Listado 2-15 está referenciando cada segundo elemento **<p>** encontrado en el documento.

> **Hágalo usted mismo:** Reemplace el código en su archivo HTML por el del Listado 2-14 y abra el archivo en su navegador. Incorpore las reglas estudiadas en el Listado 2-15 dentro del archivo **misestilos.css** para comprobar su funcionamiento.

Usando este método de referencia podemos, por supuesto, seleccionar cualquier hijo que necesitemos cambiando el número de índice. Por ejemplo, la siguiente regla tendrá impacto sólo sobre el último elemento **<p>** de nuestra plantilla:

```
p:nth-child(4){
  background: #999999;
}
```

Listado 2-16. *Pseudo clase nth-child().*

Como seguramente se habrá dado cuenta, es posible asignar estilos a todos los elementos creando una regla para cada uno de ellos:

```
*{
  margin: 0px;
}

p:nth-child(1){
  background: #999999;
}
p:nth-child(2){
  background: #CCCCCC;
}
p:nth-child(3){
  background: #999999;
}
p:nth-child(4){
  background: #CCCCCC;
}
```

Listado 2-17. *Creando una lista con la pseudo clase nth-child().*

La primera regla del Listado 2-17 usa el selector universal **`*`** para asignar el mismo estilo a cada elemento del documento. Este nuevo selector representa cada uno de los elementos en el cuerpo del documento y es útil cuando necesitamos establecer ciertas reglas básicas. En este caso, configuramos el margen de todos los elementos en 0 pixeles para evitar espacios en blanco o líneas vacías como las creadas por el elemento **`<p>`** por defecto.

En el resto del código del Listado 2-17 usamos la pseudo clase **`nth-child()`** para generar un menú o lista de opciones que son diferenciadas claramente en la pantalla por

Hágalo usted mismo: Copie el último código dentro del archivo CSS y abra el documento HTML en su navegador para comprobar el efecto.

Para agregar más opciones al menú, podemos incorporar nuevos elementos **`<p>`** en el código HTML y nuevas reglas con la pseudo clase **`nth-child()`** usando el número de índice adecuado. Sin embargo, esta aproximación genera mucho código y resulta imposible de aplicar en sitios webs con contenido dinámico. Una alternativa para obtener el mismo resultado es aprovechar las palabras clave **`odd`** y **`even`** disponibles para esta pseudo clase:

```
*{
  margin: 0px;
}
p:nth-child(odd){
  background: #999999;
}
p:nth-child(even){
  background: #CCCCCC;
}
```

Listado 2-18. *Aprovechando las palabras clave odd y even.*

Ahora solo necesitamos dos reglas para crear la lista completa. Incluso si más adelante agregamos otras opciones, los estilos serán asignados automáticamente a cada una de ellas de acuerdo a su posición. La palabra clave **`odd`** para la pseudo clase **`nth-child()`** afecta los elementos **`<p>`** que son hijos de otro elemento y tienen un índice impar. La palabra clave **`even`**, por otro lado, afecta a aquellos que tienen un índice par.

Existen otras importantes pseudo clases relacionadas con esta última, como **`first-child`**, **`last-child`** y **`only-child`**, algunas de ellas recientemente incorporadas. La pseudo clase **`first-child`** referencia solo el primer hijo, **`last-child`** referencia solo el último hijo, y **`only-child`** afecta un elemento siempre y cuando sea el único hijo disponible. Estas pseudo clases en particular no requieren palabras clave o parámetros, yson implementadas como en el siguiente ejemplo:

```
*{
  margin: 0px;
}
p:last-child{
  background: #999999;
}
```

Listado 2-19. *Usando last-child para modificar solo el último elemento <p> de la lista.*

Otra importante pseudo clase llamada **`not()`** es utilizada realizar una negación:

```
:not(p){
  margin: 0px;
}
```

Listado 2-20. *Aplicando estilos a cada elemento, excepto <p>.*

La regla del Listado 2-20 asignará un margen de 0 pixeles a cada elemento del documento excepto los elementos **`<p>`**. A diferencia del selector universal utilizado previamente, la pseudo clase **`not()`** nos permite declarar una excepción. Los estilos en la regla creada con esta pseudo clase serán asignados a todo elemento excepto aquellos incluidos en la referencia entre

paréntesis. En lugar de la palabra clave de un elemento podemos usar cualquier otra referencia que deseemos. En el próximo listado, por ejemplo, todos los elementos serán afectados excepto aquellos con el valor **mitexto2** en el atributo **class**:

```
:not(.mitexto2){
  margin: 0px;
}
```

Listado 2-21. *Excepción utilizando el atributo class.*

Cuando aplicamos la última regla al código HTML del Listado 2-14 el navegador asigna los estilos por defecto al elemento **<p>** identificado con el atributo **class** y el valor **mitexto2** y provee un margen de 0 pixeles al resto.

Nuevos selectores

Hay algunos selectores más que fueron agregados o que ahora son considerados parte de CSS3 y pueden ser útiles para nuestros diseños. Estos selectores usan los símbolos **>**, **+** y **~** para especificar la relación entre elementos.

```
div > p.mitexto2{
  color: #990000;
}
```

Listado 2-22. *Selector >.*

El selector **>** está indicando que el elemento a ser afectado por la regla es el elemento de la derecha cuando tiene al de la izquierda como su padre. La regla en el Listado 2-22 modifica los elementos **<p>** que son hijos de un elemento ** **. En este caso, fuimos bien específicos yreferenciamos solamente el elemento **<p>** con el valor **mitexto2** en su atributo **class**.

El próximo ejemplo construye un selector utilizando el símbolo **+**. Este selector referencia al elemento de la derecha cuando es inmediatamente precedido por el de la izquierda. Ambos elementos deben compartir el mismo padre:

```
p.mitexto2 + p{
  color: #990000;
}
```

Listado 2-23. *Selector +.*

La regla del Listado 2-23 afecta al elemento **<p>** que se encuentra ubicado luego de otro elemento **<p>** identificado con el valor **mitexto2** en su atributo **class**. Si abre en su navegador el archivo HTML con el código del Listado 2-14, el texto en el tercer elemento **<p>** aparecerá en la pantalla en color rojo debido a que este elemento **<p>** en particular está posicionado inmediatamente después del elemento **<p>** identificado con el valor **mitexto2** en su atributo **class**.

El último selector que estudiaremos es el construido con el símbolo **~**. Este selector es similar al anterior pero el elemento afectado no necesita estar precediendo de inmediato al elemento de la izquierda. Además, más de un elemento puede ser afectado:

```
p.mitexto2 ~ p{
  color: #990000;
}
```

Listado 2-24. *Selector ~.*

La regla del Listado 2-24 afecta al tercer y cuarto elemento **<p>** de nuestra plantilla de ejemplo. El estilo será aplicado a todos los elementos **<p>** que son hermanos y se encuentran luego del elemento **<p>** identificado con el valor **mitexto2** en su atributo **class**. No importa si otros elementos se encuentran intercalados, los elementos **<p>** en la tercera y cuarta posición aún serán afectados. Puede verificar esto último insertando un elemento **mitexto** luego del elemento **<p>** que tiene el valor **mitexto2** en su atributo **class**. A pesar de este cambio solo los elementos **<p>** serán modificados por esta regla.

2.4 Aplicando CSS a nuestra plantilla

Como aprendimos más temprano en este mismo capítulo, todo elemento estructural es considerado una caja y la estructura completa es presentada como un grupo de cajas. Las cajas agrupadas constituyen lo que es llamado un Modelo de Caja.

Siguiendo con los conceptos básicos de CSS, vamos a estudiar lo que es llamado el Modelo de Caja Tradicional. Este modelo has sido implementado desde la primera versión de CSS y es actualmente soportado por cada navegador en el mercado, lo que lo ha convertido en un estándar para el diseño web.

Todo modelo, incluso aquellos aún en fase experimental, pueden ser aplicados a la misma estructura HTML, pero esta estructura debe ser preparada para ser afectada por estos estilos de forma adecuada. Nuestros documentos HTML deberán ser adaptados al modelo de caja seleccionado.

> **IMPORTANTE:** El Modelo de Caja Tradicional presentado posteriormente no es una incorporación de HTML5, pero es introducido en este libro por ser el único disponible en estos momentos y posiblemente el que continuará siendo utilizado en sitios webs desarrollados en HTML5 durante los próximos años. Si usted ya conoce cómo implementarlo, siéntase en libertad de obviar esta parte del capítulo.

2.5 Modelo de caja tradicional

Todo comenzó con tablas. Las tablas fueron los elementos que sin intención se volvieron la herramienta ideal utilizada por desarrolladores para crear y organizar cajas de contenido en la pantalla. Este puede ser considerado el primer modelo de caja de la web. Las cajas eran creadas expandiendo celdas y combinando filas de celdas, columnas de celdas y tablas enteras, unas sobre otras o incluso anidadas. Cuando los sitios webs crecieron y se volvieron más y más complejos esta práctica comenzó a presentar serios problemas relacionados con el tamaño y el mantenimiento del código necesario para crearlos.

Estos problemas iniciales hicieron necesario lo que ahora vemos como una práctica natural: la división entre estructura y presentación. Usando etiquetas **`<div>`** y estilos CSS fue posible reemplazar la función de tablas y efectivamente separar la estructura HTML de la presentación. Con elementos **`<div>`** y CSS podemos crear cajas en la pantalla, posicionar estas cajas a un lado o a otro y darles un tamaño, color o borde específico entre otras características. CSS provee propiedades específicas que nos permiten organizar las cajas acorde a nuestros deseos. Estas propiedades son lo suficientemente poderosas como para crear un modelo de caja que se transformó en lo que hoy conocemos como Modelo de Caja Tradicional.

Algunas deficiencias en este modelo mantuvieron a las tablas vivas por algún tiempo, pero los principales desarrolladores, influenciados por el suceso de las implementaciones Ajax y una cantidad enorme de nuevas aplicaciones interactivas, gradualmente volvieron a las etiquetas **`<div>`** y estilos CSS en un estándar. Finalmente el Modelo de Caja Tradicional fue adoptado a gran escala.

Plantilla

En el Capítulo 1 construimos una plantilla HTML5. Esta plantilla tiene todos los elementos necesarios para proveer estructura a nuestro documento, pero algunos detalles deben ser agregados para aplicar los estilos CSS y el Modelo de Caja Tradicional.

Este modelo necesita agrupar cajas juntas para ordenarlas horizontalmente. Debido a que el contenido completo del cuerpo es creado a partir de cajas, debemos agregar un elemento **`<div>`** para agruparlas, centrarlas y darles un tamaño específico.

La nueva plantilla lucirá de este modo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, JavaScript">
  <title>Este texto es el título del documento</title>
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
<div id="agrupar">
  <header id="cabecera">
    <h1>Este es el título principal del sitio web</h1>
  </header>
```

```
    <nav id="menu">
      <ul>
        <li>principal</li>
        <li>fotos</li>
        <li>videos</li>
        <li>contacto</li>
      </ul>
    </nav>
    <section id="seccion">
      <article>
        <header>
          <hgroup>
            <h1>Título del mensaje uno</h1>
            <h2>Subtítulo del mensaje uno</h2>
          </hgroup>
          <time datetime="2011-12-10" pubdate>publicado 10-12-2011
          </time>
        </header>
        Este es el texto de mi primer mensaje
        <figure>
          <img src="http://minkbooks.com/content/myimage.jpg">
          <figcaption>
            Esta es la imagen del primer mensaje
          </figcaption>
        </figure>
        <footer>
          <p>comentarios (0)</p>
        </footer>
      </article>
      <article>
        <header>
          <hgroup>
            <h1>Título del mensaje dos</h1>
            <h2>Subtítulo del mensaje dos</h2>
          </hgroup>
          <time datetime="2011-12-15" pubdate>publicado 15-12-2011
          </time>
        </header>
        Este es el texto de mi segundo mensaje
        <footer>
          <p>comentarios (0)</p>
        </footer>
      </article>
    </section>
    <aside id="columna">
      <blockquote>Mensaje número uno</blockquote>
      <blockquote>Mensaje número dos</blockquote>
    </aside>
    <footer id="pie">
      Derechos Reservados &copy; 2010-2011
    </footer>
  </div>
  </body>
  </html>
```

Listado 2-25. *Nueva plantilla HTML5 lista para estilos CSS.*

El Listado 2-25 provee una nueva plantilla lista para recibir los estilos CSS. Dos cambios importantes pueden distinguirse al comparar este código con el del Listado 1-18 del Capítulo 1. El primero es que ahora varias etiquetas fueron identificadas con los atributos **`id`** y **`class`**. Esto significa que podemos referenciar un elemento específico desde las reglas CSS con el valor de su atributo **`id`** o podemos modificar varios elementos al mismo tiempo usando el valor de su atributo **`class`**.

El segundo cambio realizado a la vieja plantilla es la adición del elemento **`<div>`** mencionado anteriormente. Este **`<div>`** fue identificado con el atributo y el valor **`id="agrupar"`**, y es cerrado al final del cuerpo con la etiqueta de cierre **`</div>`**. Este

elemento se encarga de agrupar todos los demás elementos permitiéndonos aplicar el modelo de caja al cuerpo y designar su posición horizontal, como veremos más adelante.

Hágalo usted mismo: Compare el código del Listado 1-18 del Capítulo 1 con el código en el Listado 2-25 de este capítulo y ubique las etiquetas de apertura y cierre del elemento **`<div>`** utilizado para agrupar al resto. También compruebe cuáles elementos se encuentran ahora identificados con el atributo **`id`** y cuáles con el atributo **`class`**. Confirme que los valores de los atributos **`id`** son únicos para cada etiqueta. También necesitará reemplazar el código en el archivo HTML creado anteriormente por el del Listado 2-25 para aplicar los siguientes estilos CSS.

Con el documento HTML finalizado es tiempo de trabajar en nuestro archivo de estilos.

Selector universal *

Comencemos con algunas reglas básicas que nos ayudarán a proveer consistencia al diseño:

```
* {
  margin: 0px;
  padding: 0px;
}
```

Listado 2-26. *Regla CSS general.*

Normalmente, para la mayoría de los elementos, necesitamos personalizar los márgenes o simplemente mantenerlos al mínimo. Algunos elementos por defecto tienen márgenes que son diferentes de cero y en la mayoría de los casos demasiado amplios. A medida que avanzamos en la creación de nuestro diseño encontraremos que la mayoría de los elementos utilizados deben tener un margen de 0 pixeles. Para evitar el tener que repetir estilos constantemente, podemos utilizar el selector universal.

La primera regla en nuestro archivo CSS, presentada en el Listado 2-26, nos asegura que todo elemento tendrá un margen interno y externo de 0 pixeles. De ahora en más solo necesitaremos modificar los márgenes de los elementos que queremos que sean mayores que cero.

Conceptos básicos: Recuerde que en HTML cada elemento es considerado como una caja. El margen (**`margin`**) es en realidad el espacio alrededor del elemento, el que se encuentra por fuera del borde de esa caja (el estilo **`padding`**, por otro lado, es el espacio alrededor del contenido del elemento pero dentro de sus bordes, como el espacio entre el título y el borde de la caja virtual formada por el elemento **`<h1>`** que contiene ese título). El tamaño del margen puede ser definido por lados específicos del elemento o todos sus lados a la vez. El estilo **`margin: 0px`** en nuestro ejemplo establece un margen 0 o nulo para cada elemento de la caja. Si el tamaño hubiese sido especificado en 5 pixeles, por ejemplo, la caja tendría un espacio de 5 pixeles de ancho en todo su contorno. Esto significa que la caja estaría separada de sus vecinas por 5 pixeles. Volveremos sobre este tema más adelante en este capítulo.

Hágalo usted mismo: Debemos escribir todas las reglas necesarias para otorgar estilo a nuestra plantilla en un archivo CSS. El archivo ya fue incluido dentro del código HTML por medio de la etiqueta **`<link>`**, por lo que lo único que tenemos que hacer es crear un archivo de texto vacío con nuestro editor de textos preferido, grabarlo con el nombre **`misestilos.css`** yluego copiar en su interior la regla del Listado 2-26 ytodas las presentadas a continuación.

Nueva jerarquía para cabeceras

En nuestra plantilla usamos elementos **`<h1>`** y **`<h2>`** para declarar títulos y subtítulos de diferentes secciones del documento. Los estilos por defecto de estos elementos se encuentran siempre muy lejos de lo que queremos y además en HTML5 podemos reconstruir la jerarquía H varias veces en cada sección (como aprendimos en el capítulo anterior). El elemento **`<h1>`**, por ejemplo, será usado varias veces en el documento, no solo para el título principal de la página web como pasaba anteriormente sino también para secciones internas, por lo que tenemos que otorgarle los estilos apropiados:

```
h1 {
  font: bold 20px verdana, sans-serif;
}
h2 {
  font: bold 14px verdana, sans-serif;
}
```

Listado 2-27. *Agregando estilos para los elementos <h1> y <h2>.*

La propiedad **font**, asignada a los elementos **<h1>** y **<h2>** en el Listado 2-27, nos permite declarar todos los estilos para el texto en una sola línea. Las propiedades que pueden ser declaradas usando **font** son: **font-style**, **font-variant**, **font-weight**, **font-size/line-height**, y **font-family** en este orden. Con estas reglas estamos cambiando el grosor, tamaño ytipo de letra del texto dentro de los elementos **<h1>** y **<h2>** a los valores que deseamos.

Declarando nuevos elementos HTML5

Otra regla básica que debemos declarar desde el comienzo es la definición por defecto de elementos estructurales de HTML5. Algunos navegadores aún no reconocen estos elementos o los tratan como elementos *inline* (en línea). Necesitamos declarar los nuevos elementos HTML5 como elementos *block* para asegurarnos de que serán tratados como regularmente se hace con elementos ** ** y de este modo construir nuestro modelo de caja:

```
header, section, footer, aside, nav, article, figure, figcaption,
hgroup{
  display: block;
}
```

Listado 2-28. *Regla por defecto para elementos estructurales de HTML5.*

Apartir de ahora, los elementos afectados por la regla del Listado 2-28 serán posicionados uno sobre otro a menos que especifiquemos algo diferente más adelante.

Centrando el cuerpo

El primer elemento que es parte del modelo de caja es siempre **<body>**. Normalmente, por diferentes razones de diseño, el contenido de este elemento debe ser posicionado horizontalmente. Siempre deberemos especificar el tamaño de este contenido, o un tamaño máximo, para obtener un diseño consistente a través de diferentes configuraciones de pantalla.

```
body {
  text-align: center;
}
```

Listado 2-29. *Centrando el cuerpo.*

Por defecto, la etiqueta **<body>** (como cualquier otro elemento *block*) tiene un valor de ancho establecido en 100%. Esto significa que el cuerpo ocupará el ancho completo de la ventana del navegador. Por lo tanto, para centrar la página en la pantalla necesitamos centrar el contenido dentro del cuerpo. Con la regla agregada en el Listado 2-29, todo lo que se encuentra dentro de **<body>** será centrado en la ventana, centrando de este modo toda la página web.

Creando la caja principal

Siguiendo con el diseño de nuestra plantilla, debemos especificar una tamaño o tamaño máximo para el contenido del cuerpo. Como seguramente recuerda, en el Listado 2-25 en este mismo capítulo agregamos un elemento ** ** a la plantilla para agrupar todas las cajas dentro del cuerpo. Este ** ** será considerado la caja principal para la construcción de nuestro modelo de caja (este es el propósito por el que lo agregamos). De este modo, modificando el tamaño de este elemento lo hacemos al mismo tiempo para todos los demás:

```
#agrupar {
  width: 960px;
  margin: 15px auto;
  text-align: left;
}
```

Listado 2-30. *Definiendo las propiedades de la caja principal.*

La regla en el Listado 2-30 está referenciando por primera vez un elemento a través del valor de su atributo **id**. El carácter **#** le está diciendo al navegador que el elemento afectado por este conjunto de estilos tiene el atributo **id** con el valor **agrupar**.

Esta regla provee tres estilos para la caja principal. El primer estilo establece un valor fijo de 960 pixeles. Esta caja tendrá siempre un ancho de 960 pixeles, lo que representa un valor común para un sitio web estos días (los valores se encuentran entre 960 y 980 pixeles de ancho, sin embargo estos parámetros cambian constantemente a través del tiempo, por supuesto).

El segundo estilo es parte de lo que llamamos el Modelo de Caja Tradicional. En la regla previa (Listado 2-29), especificamos que el contenido del cuerpo sería centrado horizontalmente con el estilo **text-align: center**. Pero esto solo afecta contenido *inline*, como textos o imágenes. Para elementos *block*, como un ** **, necesitamos establecer un valor específico para sus márgenes que los adapta automáticamente al tamaño de su elemento padre. La propiedad **margin** usada para este propósito puede tener cuatro valores: superior, derecho, inferior, izquierdo, en este orden. Esto significa que el primer valor declarado en el estilo representa el margen de la parte superior del elemento, el segundo es el margen de la derecha, y así sucesivamente. Sin embargo, si solo escribimos los primeros dos parámetros, el resto tomará los mismos
valores. En nuestro ejemplo estamos usando esta técnica.

En el Listado 2-30, el estilo **margin: 15px auto** asigna 15 pixeles al margen superior e inferior del elemento ** ** que está afectando y declara como automático el tamaño de los márgenes de izquierda y derecha (los dos valores declarados son usados para definir los cuatro márgenes). De esta manera, habremos generado un espacio de 15 pixeles en la parte superior e inferior del cuerpo y los espacios a los laterales (margen izquierdo y derecho) serán calculados automáticamente de acuerdo al tamaño del cuerpo del documento y el elemento ** **, efectivamente centrando el contenido en pantalla.

La página web ya está centrada y tiene un tamaño fijo de 960 pixeles. Lo próximo que necesitamos hacer es prevenir un problema que ocurre en algunos navegadores. La propiedad **text-align** es hereditaria. Esto significa que todos los elementos dentro del cuerpo y su contenido serán centrados, no solo la caja principal. El estilo asignado a **<body>** en el Listado 2-29 será asignado a cada uno de sus hijos. Debemos retornar este estilo a su valor por defecto para el resto del documento. El tercer y último estilo incorporado en la regla del Listado 2-30 (**text-align: left**) logra este propósito. El resultado final es que el contenido del cuerpo es centrado pero el contenido de la caja principal (el ** ** identificado como **agrupar**) es alineado nuevamente hacia la izquierda, por lo tanto todo el resto del código HTML dentro de esta caja hereda este estilo.

> **Hágalo usted mismo:** Si aún no lo ha hecho, copie cada una de las reglas listadas hasta este punto dentro de un archivo de texto vacío llamado **misestilos.css**. Este archivo debe estar ubicado en el mismo directorio (carpeta) que el archivo HTML con el código del Listado 2-25. Al terminar, deberá contar con dos archivos, uno con el código HTML y otro llamado **misestilos.css** con todos los estilos CSS estudiados desde el Listado 2-26.Abra el archivo HTML en su navegador y en la pantalla podrá notar la caja creada.

La cabecera

Continuemos con el resto de los elementos estructurales. Siguiendo la etiqueta de apertura del ** ** principal se encuentra el primer elemento estructural de HTML5: **<header>**. Este elemento contiene el título principal de nuestra página web y estará ubicado en la parte superior de la pantalla. En nuestra plantilla, **<header>** fue identificado con el atributo **i d** y el valor **cabecera**.

Como ya mencionamos, cada elemento *block*, así como el cuerpo, por defecto tiene un valor de ancho del 100%. Esto significa que el elemento ocupará todo el espacio horizontal disponible. En el caso del cuerpo, ese espacio es el ancho total de la pantalla visible (la ventana del navegador), pero en el resto de los elementos el espacio máximo disponible estará determinado por el ancho de su elemento padre. En nuestro ejemplo, el espacio máximo disponible para los elementos dentro de la caja principal será de 960 pixeles, porque su padre es la caja principal la cual fue previamente configurada con este tamaño.

```
#cabecera {
  background: #FFFBB9;
  border: 1px solid #999999;
  padding: 20px;
}
```

Listado 2-31. *Agregando estilos para <header>.*

Debido a que **<header>** ocupará todo el espacio horizontal disponible en la caja principal y será tratado como un elemento *block* (y por esto posicionada en la parte superior de la página), lo único que resta por hacer es asignar estilos que nos permitirán reconocer el elemento cuando es presentado en pantalla. En la regla mostrada en el Listado 2-31 le otorgamos a **<header>** un fondo amarillo, un borde sólido de 1 pixel y un margen interior de 20 pixeles usando la propiedad **padding**.

Barra de navegación

Siguiendo al elemento **<header>** se encuentra el elemento **<nav>**, el cual tiene el propósito de proporcionar ayuda para la navegación. Los enlaces agrupados dentro de este elemento representarán el menú de nuestro sitio web. Este menú será una simple barra ubicada debajo de la cabecera. Por este motivo, del mismo modo que el elemento **<header>**, la mayoría de los estilos que necesitamos para posicionar el elemento **<nav>** ya fueron asignados: **<nav>** es un elemento *block* por lo que será ubicado debajo del elemento previo, su ancho por defecto será 100% por lo que será tan ancho como su padre (el ** ** principal), y (también por defecto) será tan alto como su contenido y los márgenes predeterminados. Por lo tanto, lo único que nos queda por hacer es mejorar su aspecto en pantalla. Esto último lo logramos agregando un fondo gris y un pequeño margen interno para separar las opciones del menú del borde del elemento:

```
#menu {
  background: #CCCCCC;
  padding: 5px 15px;
}
#menu li {
  display: inline-block;
  list-style: none;
  padding: 5px;
  font: bold 14px verdana, sans-serif;
}
```

Listado 2-32. *Agregando estilos para <nav>.*

En el Listado 2-32, la primera regla referencia al elemento **<nav>** por su atributo **id**, cambia su color de fondo y agrega márgenes internos de **5px** y **15px** con la propiedad **padding**.

> **Conceptos básicos:** La propiedad **padding** trabaja exactamente como **margin**. Cuatro valores pueden ser especificados: superior, derecho, inferior, izquierdo, en este orden. Si solo declaramos un valor, el mismo será asignado para todos los espacios alrededor del contenido del elemento. Si en cambio especificamos dos valores, entonces el primero será asignado como margen interno de la parte superior e inferior del contenido y el segundo valor será asignado al margen interno de los lados, izquierdo y derecho.

Dentro de la barra de navegación hay una lista creada con las etiquetas **<ul>** y **<li>**. Por defecto, los ítems de una lista son posicionados unos sobre otros. Para cambiar este comportamiento y colocar cada opción del menú una al lado de la otra, referenciamos los elementos **<li>** dentro de este elemento **<nav>** en particular usando el selector **#menu li**, y luego asignamos a todos ellos el estilo **display: inline-block** para convertirlos en lo que se llama cajas *inline*. Adiferencia de los elementos *block*, los elementos afectados por el parámetro **inline-block** estandarizado en CSS3 no generan ningún salto de línea pero nos permiten tratarlos como elementos block y así declarar un valor de ancho determinado. Este parámetro también ajusta el tamaño del elemento de acuerdo con su contenido cuando el valor del ancho no fue especificado.

En esta última regla también eliminamos el pequeño gráfico generado por defecto por los navegadores delante de cada opción del listado utilizando la propiedad **list-style**.

Section y aside

Los siguientes elementos estructurales en nuestro código son dos cajas ordenadas horizontalmente. El Modelo de Caja Tradicional es construido sobre estilos CSS que nos permiten especificar la posición de cada caja. Usando la propiedad **float** podemos posicionar estas cajas del lado izquierdo o derecho de acuerdo a nuestras necesidades. Los elementos que utilizamos en nuestra plantilla HTML para crear estas cajas son **<section>** y **<aside>**, cada uno identificado con el atributo **id** ylos valores **seccion** y **columna** respectivamente.

```
#seccion {
  float: left;
  width: 660px;
  margin: 20px;
}
#columna {
  float: left;
  width: 220px;
```

```
  margin: 20px 0px;
  padding: 20px;
  background: #CCCCCC;
}
```

Listado 2-33. *Creando dos columnas con la propiedad float.*

La propiedad de CSS **`float`** es una de las propiedades más ampliamente utilizadas para aplicar el Modelo de Caja Tradicional. Hace que el elemento flote hacia un lado o al otro en el espacio disponible. Los elementos afectados por **`float`** actúan como elementos *block* (con la diferencia de que son ubicados de acuerdo al valor de esta propiedad y no el flujo normal del documento). Los elementos son movidos a izquierda o derecha en el área disponible, tanto como sea posible, respondiendo al valor de **`float`**.

Con las reglas del Listado 2-33 declaramos la posición de ambas cajas y sus respectivos tamaños, generando así las columnas visibles en la pantalla. La propiedad **`float`** mueve la caja al espacio disponible del lado especificado por su valor, **`width`** asigna un tamaño horizontal y **`margin`**, por supuesto, declara el margen del elemento.

Afectado por estos valores, el contenido del elemento **`<section>`** estará situado a la izquierda de la pantalla con un tamaño de 660 pixeles, más 40 pixeles de margen, ocupando un espacio total de 700 pixeles de ancho.

La propiedad **`float`** del elemento **`<aside>`** también tiene el valor **`left`** (izquierda). Esto significa que la caja generada será movida al espacio disponible a su izquierda. Debido a que la caja previa creada por el elemento **`<section>`** fue también movida a la izquierda de la pantalla, ahora el espacio disponible será solo el que esta caja dejó libre. La nueva caja quedará ubicada en la misma línea que la primera pero a su derecha, ocupando el espacio restante en la línea, creando la segunda columna de nuestro diseño.

El tamaño declarado para esta segunda caja fue de 220 pixeles. También agregamos un fondo gris y configuramos un margen interno de 20 pixeles. Como resultado final, el ancho de esta caja será de 220 pixeles más 40 pixeles agregados por la propiedad **`padding`** (los márgenes de los lados fueron declarados a **`0px`**).

> **Conceptos básicos:** El tamaño de un elemento y sus márgenes son agregados para obtener el valor real ocupado en pantalla. Si tenemos un elemento de 200 pixeles de ancho y un margen de 10 pixeles a cada lado, el área real ocupada por el elemento será de 220 pixeles. El total de 20 pixeles del margen es agregado a los 200 pixeles del elemento y el valor final es representado en la pantalla. Lo mismo pasa con las propiedades **`padding`** y **`border`**. Cada vez que agregamos un borde a un elemento o creamos un espacio entre el contenido y el borde usando **`padding`** esos valores serán agregados al ancho del elemento para obtener el valor real cuando el elemento es mostrado en pantalla. Este valor real es calculado con la fórmula: **`tamaño + márgenes + márgenes internos + bordes`**.
>
> **Hágalo usted mismo:** Lea el código del Listado 2-25. Controle cada regla CSS creada hasta el momento y busque en la plantilla los elementos HTML correspondientes a cada una de ellas. Siga las referencias, por ejemplo las claves de los elementos (como **`h1`**) y los atributos **`id`** (como **`cabecera`**), para entender cómo trabajan las referencias y cómo los estilos son asignados a cada elemento.

Footer

Para finalizar la aplicación del Modelo de Caja Tradicional, otra propiedad CSS tiene que ser aplicada al elemento **`<footer>`**. Esta propiedad devuelve al documento su flujo normal y nos permite posicionar **`<footer>`** debajo del último elemento en lugar de a su lado:

```
#pie {
  clear: both;
  text-align: center;
  padding: 20px;
  border-top: 2px solid #999999;
}
```

Listado 2-34. *Otorgando estilos a <footer> y recuperando el normal flujo del documento.*

La regla del Listado 2-34 declara un borde de 2 pixeles en la parte superior de **`<footer>`**, un margen interno (**`padding`**) de 20 pixeles, y centra el texto dentro del elemento. Así mismo, restaura el normal flujo del documento con la propiedad **`clear`**. Esta propiedad simplemente restaura las condiciones normales del área ocupada por el elemento, no permitiéndole posicionarse adyacente a una caja flotante. El valor usualmente utilizado es **`both`**, el cual significa que ambos lados del elemento serán restaurados y el elemento seguirá el flujo normal (este elemento ya no es flotante como los anteriores). Esto, para un elemento

block, quiere decir que será posicionado debajo del último elemento, en una nueva línea.

La propiedad **clear** también empuja los elementos verticalmente, haciendo que las cajas flotantes ocupen un área real en la pantalla. Sin esta propiedad, el navegador presenta el documento en pantalla como si los elementos flotantes no existieran ylas cajas se superponen.

Figura 2-2. Representación visual del modelo de caja tradicional.

Cuando tenemos cajas posicionadas una al lado de la otra en el Modelo de Caja Tradicional siempre necesitamos crear un elemento con el estilo **clear: both** para poder seguir agregando otras cajas debajo de un modo natural. La Figura 2-2 muestra una representación visual de este modelo con los estilos básicos para lograr la correcta disposición en pantalla.

Los valores **left** (izquierda) y **right** (derecha) de la propiedad **float** no significan que las cajas deben estar necesariamente posicionadas del lado izquierdo o derecho de la ventana. Lo que los valores hacen es volver flotante ese lado del elemento, rompiendo el flujo normal del documento. Si el valor es **left**, por ejemplo, el navegador tratará de posicionar el elemento del lado izquierdo en el espacio disponible. Si hay espacio disponible luego de otro elemento, este nuevo elemento será situado a su derecha, porque su lado izquierdo fue configurado como flotante. El elemento flota hacia la izquierda hasta que encuentra algo que lo bloquea, como otro elemento o el borde de su elemento padre. Esto es importante cuando queremos crear varias columnas en la pantalla. En este caso cada columna tendrá el valor **left** en la propiedad **float** para asegurar que cada columna estará continua a la otra en el orden correcto. De este modo, cada columna flotará hacia la izquierda hasta que es bloqueada por otra columna o el borde del elemento padre.

Últimos toques

Lo único que nos queda por hacer es trabajar en el diseño del contenido. Para esto, solo necesitamos configurar los pocos elementos HTML5 restantes:

```
article {
  background: #FFFBCC;
  border: 1px solid #999999;
  padding: 20px;
  margin-bottom: 15px;
}
article footer {
  text-align: right;
}
time {
  color: #999999;
}
figcaption {
  font: italic 14px verdana, sans-serif;
}
```

Listado 2-35. *Agregando los últimos toques a nuestro diseño básico.*

La primera regla del Listado 2-35 referencia todos los elementos **<article>** y les otorga algunos estilos básicos (color de fondo, un borde sólido de 1 pixel, margen interno y margen inferior). El margen inferior de 15 pixeles tiene el propósito de separar un elemento **<article>** del siguiente verticalmente.

Cada elemento **<article>** cuenta también con un elemento **<footer>** que muestra el número de comentarios recibidos. Para referenciar un elemento **<footer>** dentro de un elemento **<article>**, usamos el selector **article footer** que significa “cada **<footer>** dentro de un **<article>** será afectado por los siguientes estilos”. Esta técnica de referencia fue aplicada aquí para alinear a la derecha el texto dentro de los elementos **<footer>** de cada **<article>**.

Al final del código en el Listado 2-35 cambiamos el color de cada elemento **<time>** y diferenciamos la descripción de la imagen (insertada con el elemento **<figcaption>**) del resto del texto usando una tipo de letra diferente.

> **Hágalo usted mismo:** Si aún no lo ha hecho, copie cada regla CSS listada en este capítulo desde el Listado 2-26, una debajo de otra, dentro del archivo **misestilos.css**, yluego abra el archivo HTML con la plantilla creada en el Listado 2-25 en su navegador. Esto le mostrará cómo funciona el Modelo de Caja Tradicional y cómo los elementos estructurales son organizados en pantalla.
>
> **IMPORTANTE:** Puede acceder a estos códigos con un solo clic desde nuestro sitio web. Visite *www.minkbooks.com*.

Box-sizing

Existe una propiedad adicional incorporada en CSS3 relacionada con la estructura y el Modelo de Caja Tradicional. La propiedad **box-sizing** nos permite cambiar cómo el espacio total ocupado por un elemento en pantalla será calculado forzando a los navegadores a incluir en el ancho original los valores de las propiedades **padding** y **border**.

Como explicamos anteriormente, cada vez que el área total ocupada por un elemento es calculada, el navegador obtiene el valor final por medio de la siguiente fórmula: **tamaño + márgenes + márgenes internos + bordes**.

Por este motivo, si declaramos la propiedad **width** igual a 100 pixeles, **margin** en 20 pixeles, **padding** en 10 pixeles y **border** en 1 pixel, el área horizontal total ocupada por el elemento será: **100+40+20+2= 162** pixeles (note que tuvimos que duplicar los valores de **margin**, **padding** y **border** en la fórmula porque consideramos que los mismos fueron asignados tanto para el lado derecho como el izquierdo).

Esto significa que cada vez que declare el ancho de un elemento con la propiedad **width**, deberá recordar que el área real para ubicar el elemento en pantalla será seguramente más grande.

Dependiendo de sus costumbres, a veces podría resultar útil forzar al navegador a incluir los valores de **padding** y **border** en el tamaño del elemento. En este caso la nueva fórmula sería simplemente: **tamaño + márgenes**.

```
div {
  width: 100px;
  margin: 20px;
  padding: 10px;
  border: 1px solid #000000;

  -moz-box-sizing: border-box;
  -webkit-box-sizing: border-box;
  box-sizing: border-box;
}
```

Listado 2-36. *Incluyendo padding y border en el tamaño del elemento.*

La propiedad **box-sizing** puede tomar dos valores. Por defecto es configurada como **content-box**, lo que significa que los navegadores agregarán los valores de **padding** y **border** al tamaño especificado por **width** y **height**. Usando el valor **border-box** en su lugar, este comportamiento es cambiado de modo que **padding** y **border** son incluidos dentro del elemento.

El Listado 2-36 muestra la aplicación de esta propiedad en un elemento ** **. Este es solo un ejemplo y no vamos a usarlo en nuestra plantilla, pero puede ser útil para algunos diseñadores dependiendo de qué tan familiarizados se encuentran con los métodos tradicionales propuestos por versiones previas de CSS.

> **IMPORTANTE:** En este momento, la propiedad **box-sizing**, al igual que otras importantes propiedades CSS3 estudiadas en próximos capítulos, se encuentra en estado experimental en algunos navegadores. Para aplicarla efectivamente a sus documentos, debe declararla con los correspondientes prefijos, como hicimos en el Listado 2-36. Los prefijos para los navegadores más comunes son los siguientes:
>
> · **-moz-** para Firefox.
>
> · **-webkit-** para Safari y Chrome. ·
>
> **-o-** para Opera.
>
> · **-khtml-** para Konqueror.
>
> · **-ms-** para Internet Explorer.
>
> · **-chrome-** específico para Google Chrome.

2.6 Referencia rápida

En HTML5 la responsabilidad por la presentación de la estructura en pantalla está más que nunca en manos de CSS. Incorporaciones y mejoras se han hecho en la última versión de CSS para proveer mejores formas de organizar documentos ytrabajar con sus elementos.

Selector de atributo y pseudo clases

CSS3 incorpora nuevos mecanismos para referenciar elementos HTML.

Selector de Atributo Ahora podemos utilizar otros atributos además de `id` y `class` para encontrar elementos en el documento y asignar estilos. Con la construcción `palabraclave[atributo=valor]`, podemos referenciar un elemento que tiene un atributo particular con un valor específico. Por ejemplo, `p[name=″texto″]` referenciará cada elemento `<p>` con un atributo llamado `name` y el valor ″`texto`″. CSS3 también provee técnicas para hacer esta referencia aún más específica. Usando las siguientes combinaciones de símbolos `^=`, `$=` y `*=` podemos encontrar elementos que comienzan con el valor provisto, elementos que terminan con ese valor y elementos que tienen el texto provisto en alguna parte del valor del atributo. Por ejemplo, `p[name^=″texto″]` será usado para encontrar elementos `<p>` que tienen un atributo llamado `name` con un valor que comienza por ″`texto`″.

Pseudo Clase :nth-child() Esta pseudo clase encuentra un hijo específico siguiendo la estructura de árbol de HTML. Por ejemplo, con el estilo `span:nth-child(2)` estamos referenciando el elemento `<span>` que tiene otros elementos `<span>` como hermanos y está localizado en la posición `2`. Este número es considerado el índice. En lugar de un número podemos usar las palabras clave `odd` y `even` para referenciar elementos con un índice impar o par respectivamente (por ejemplo, `span:nth-child(odd)`).

Pseudo Clase :first-child Esta pseudo clase es usada para referenciar el primer hijo, similar a `:nth-child(1)`.

Pseudo Clase :last-child Esta pseudo clase es usada para referenciar el último hijo.

Pseudo Clase :only-child Esta pseudo clase es usada para referenciar un elemento que es el único hijo disponible de un mismo elemento padre.

Pseudo Clase :not() Esta pseudo clase es usada para referenciar todo elemento excepto el declarado entre paréntesis.

Selectores

CSS3 también incorpora nuevos selectores que ayudan a llegar a elementos difíciles de referenciar utilizando otras técnicas.

Selector > Este selector referencia al elemento de la derecha cuando tiene el elemento de la izquierda como padre. Por ejemplo, `div > p` referenciará cada elemento `<p>` que es hijo de un elemento `<div>`.

Selector + Este selector referencia elementos que son hermanos. La referencia apuntará al elemento de la derecha cuando es inmediatamente precedido por el de la izquierda. Por ejemplo, `span + p` afectará a los elementos `<p>` que son hermanos y están ubicados luego de un elemento `<span>`.

Selector ~ Este selector es similar al anterior, pero en este caso el elemento de la derecha no tiene que estar ubicado inmediatamente después del de la izquierda.

Capítulo 3
Propiedades CSS3

3.1 Las nuevas reglas

La web cambió para siempre cuando unos años atrás nuevas aplicaciones desarrolladas sobre implementaciones Ajax mejoraron el diseño y la experiencia de los usuarios. La versión 2.0, asignada a la web para describir un nuevo nivel de desarrollo, representó un cambio no solo en la forma en que la información era transmitida sino también en cómo los sitios web y nuevas aplicaciones eran diseñados yconstruidos.

Los códigos implementados en esta nueva generación de sitios web pronto se volvieron estándar. La innovación se volvió tan importante para el éxito de cualquier proyecto en Internet que programadores desarrollaron librerías completas para superar las limitaciones ysatisfacer los nuevos requerimientos de los diseñadores.

La falta de soporte por parte de los navegadores era evidente, pero la organización responsable de los estándares web no tomó las tendencias muy seriamente e intentó seguir su propio camino. Afortunadamente, algunas mentes brillantes siguieron desarrollando nuevos estándares en paralelo y pronto HTML5 nació. Luego del retorno de la calma (y algunos acuerdos de por medio), la integración entre HTML, CSS y Javascript bajo la tutela de HTML5 fue como el caballero bravo y victorioso que dirige

las tropas hacia el palacio enemigo.

A pesar de la reciente agitación, esta batalla comenzó mucho tiempo atrás, con la primera especificación de la tercera versión de CSS. Cuando finalmente, alrededor del año 2005, esta tecnología fue oficialmente considerada estándar, CSS estaba listo para proveer las funciones requeridas por desarrolladores (aquellas que programadores habían creado desde años atrás usando códigos Javascript complicados de implementar y no siempre compatibles).

En este capítulo vamos a estudiar las contribuciones hechas por CSS3 a HTML5 y todas las propiedades que simplifican la vida de diseñadores y programadores.

CSS3 se vuelve loco

CSS fue siempre sobre estilo, pero ya no más. En un intento por reducir el uso de código Javascript y para estandarizar funciones populares, CSS3 no solo cubre diseño y estilos web sino también forma y movimiento. La especificación de CSS3 es presentada en módulos que permiten a la tecnología proveer una especificación estándar por cada aspecto involucrado en la presentación visual del documento. Desde esquinas redondeadas ysombras hasta transformaciones yreposicionamiento de los elementos ya presentados en pantalla, cada posible efecto aplicado previamente utilizando Javascript fue cubierto. Este nivel de cambio convierte CSS3 en una tecnología prácticamente inédita comparada con versiones anteriores.

Cuando la especificación de HTML5 fue escrita considerando CSS a cargo del diseño, la mitad de la batalla contra el resto de las especificaciones propuesta había sido ganada.

Plantilla

Las nuevas propiedades CSS3 son extremadamente poderosas y deben ser estudiadas una por una, pero para facilitar su aprendizaje vamos a aplicar todas ellas sobre la misma plantilla. Por este motivo comenzaremos por crear un documento HTML sencillo con algunos estilos básicos:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Nuevos Estilos CSS3</title>
  <link rel="stylesheet" href="nuevocss3.css">
</head>
<body>
<header id="principal">
  <span id="titulo">Estilos CSS Web 2.0</span>
</header>
</body>
</html>
```

Listado 3-1. *Una plantilla simple para probar nuevas propiedades.*

Nuestro documento solo tiene una sección con un texto breve en su interior. El elemento **`<header>`** usado en la plantilla podría ser reemplazado por **`<div>`**, **`<nav>`**, **`<section>`** o cualquier otro elemento estructural de acuerdo a la ubicación en el diseño y a su función. Luego de aplicar los estilos, la caja generada con el código del ejemplo del Listado 3-1 lucirá como una cabecera, por consiguiente decidimos usar **`<header>`** en este caso.

Debido a que el elemento **`<font>`** se encuentra en desuso en HTML5, los elementos usados para mostrar texto son normalmente **`<span>`** para líneas cortas y **`<p>`** para párrafos, entre otros. Por esta razón el texto en nuestra plantilla fue insertado usando etiquetas **`<span>`**.

> **Hágalo usted mismo:** Use el código provisto en el Listado 3-1 como la plantilla para este capítulo. Necesitará además crear un nuevo archivo CSS llamado **`nuevocss3.css`** para almacenar los estilos estudiados de aquí en adelante.

Los siguientes son los estilos básicos requeridos por nuestro documento HTML:

```
body {
  text-align: center;
}
#principal {
  display: block;
```

```
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;
}
#titulo {
  font: bold 36px verdana, sans-serif;
}
```

Listado 3-2. *Reglas básicas CSS con las que comenzar.*

No hay nada nuevo en las reglas del Listado 3-2, solo los estilos necesarios para dar forma a la plantilla y crear una caja ancha, posicionada en el centro de la ventana, con un fondo gris, un borde y un texto grande en su interior que dice “Estilos CSS Web 2.0”.

Una de las cosas que notará sobre esta caja cuando sea mostrada en pantalla es que sus esquinas son rectas. Esto no es algo que nos agrade, ¿verdad? Puede ser un factor psicológico o no, lo cierto es que a casi nadie en este negocio le agradan las esquinas rectas. Por lo tanto, lo primero que haremos será cambiar este aspecto.

Border-radius

Por muchos años diseñadores han sufrido intentando lograr el efecto de esquinas redondeadas en las cajas de sus páginas web. El proceso era casi siempre frustrante y extenuante. Todos lo padecieron alguna vez. Si mira cualquier presentación en video de las nuevas características incorporadas en HTML5, cada vez que alguien habla sobre las propiedades de CSS3 que hacen posible generar fácilmente esquinas redondeadas, la audiencia enloquece. Esquinas redondeadas eran esa clase de cosas que nos hacían pensar: “debería ser fácil hacerlo”. Sin embargo nunca lo fue.

Esta es la razón por la que, entre todas las nuevas posibilidades e increíbles propiedades incorporadas en CSS3, la que exploraremos en primera instancia es **`border-radius`**:

```
body {
  text-align: center;
}
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
  border-radius: 20px;
}
#titulo {
  font: bold 36px verdana, sans-serif;
}
```

Listado 3-3. *Generando esquinas redondeadas.*

La propiedad **`border-radius`** en este momento es experimental por lo que debemos usar los prefijos **`-moz-`** y **`-webkit`** para que funcionen en navegadores basados en motores Gecko y WebKit, como Firefox, Safari y Google Chrome (los prefijos fueron estudiados y aplicados en el Capítulo 2). Si todas las esquinas tienen la misma curvatura podemos utilizar un solo valor. Sin embargo, como ocurre con las propiedades **`margin`** y **`padding`**, podemos también declarar un valor diferente por cada una:

```
body {
```

```
  text-align: center;
}
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-border-radius: 20px 10px 30px 50px;
  -webkit-border-radius: 20px 10px 30px 50px;
  border-radius: 20px 10px 30px 50px;
}
#titulo {
  font: bold 36px verdana, sans-serif;
}
```

Listado 3-4. *Diferentes valores para cada esquina.*

Como puede ver en el Listado 3-4, los cuatro valores asignados a la propiedad **`border-radius`** representan diferentes ubicaciones. Recorriendo la caja en dirección de las agujas del reloj, los valores se aplicarán en el siguiente orden: esquina superior izquierda, esquina superior derecha, esquina inferior derecha y esquina inferior izquierda. Los valores son siempre dados en dirección de las agujas del reloj, comenzando por la esquina superior izquierda.

Al igual que con **`margin`** o **`padding`**, **`border-radius`** puede también trabajar solo con dos valores. El primer valor será asignado a la primera y tercera equina (superior izquierda, inferior derecha), y el segundo valor a la segunda y cuarta esquina (superior derecha, inferior izquierda).

También podemos dar forma a las esquinas declarando un segundo grupo de valores separados por una barra. Los valores a la izquierda de la barra representarán el radio horizontal mientras que los valores a la derecha representan el radio vertical. La combinación de estos valores genera una elipsis:

```
body {
  text-align: center;
}
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;
  -moz-border-radius: 20px / 10px;
  -webkit-border-radius: 20px / 10px;
  border-radius: 20px / 10px;
}
#titulo {
  font: bold 36px verdana, sans-serif;
}
```

Listado 3-5. *Esquinas elípticas.*

Hágalo usted mismo: Copie dentro del archivo CSS llamado **`nuevocss3.css`** los estilos que quiera probar y abra el archivo HTML generado con el Listado 3-1 en su navegador para comprobar los resultados.

Box-shadow

Ahora que finalmente contamos con la posibilidad de generar bonitas esquinas para nuestras cajas podemos arriesgarnos con algo más. Otro muy buen efecto, que había sido extremadamente complicado de lograr hasta este momento, es sombras. Por

años diseñadores han combinado imágenes, elementos y algunas propiedades CSS para generar sombras. Gracias a CSS3 y a la nueva propiedad **box-shadow** podremos aplicar sombras a nuestras cajas con solo una simple línea de código:

```
body {
  text-align: center;
}
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
  border-radius: 20px;

  -moz-box-shadow: rgb(150,150,150) 5px 5px;
  -webkit-box-shadow: rgb(150,150,150) 5px 5px;
  box-shadow: rgb(150,150,150) 5px 5px;
}
#titulo {
  font: bold 36px verdana, sans-serif;
}
```

Listado 3-6. *Aplicando sombra a nuestra caja.*

La propiedad **box-shadow** necesita al menos tres valores. El primero, que puede ver en la regla del Listado 3-6, es el color. Este valor fue construido aquí utilizando la función **rgb()** y números decimales, pero podemos escribirlo en números hexadecimales también, como hicimos previamente para otros parámetros en este libro.

Los siguientes dos valores, expresados en pixeles, establecen el desplazamiento de la sombra. Este desplazamiento puede ser positivo o negativo. Los valores indican, respectivamente, la distancia horizontal y vertical desde la sombra al elemento. Valores negativos posicionarán la sombra a la izquierda y arriba del elemento, mientras que valores positivos crearán la sombra a la derecha y debajo del elemento. Valores de 0 o nulos posicionarán la sombra exactamente detrás del elemento, permitiendo la posibilidad de crear un efecto difuminado a todo su alrededor.

> **Hágalo usted mismo:** Para probar los diferentes parámetros y posibilidades con los que contamos para asignar una sombra a una caja, copie el código del Listado 3-6 dentro del archivo CSS y abra el archivo HTML con la plantilla del Listado 3-1 en su navegador. Puede experimentar cambiando los valores de la propiedad **box-shadow** y puede usar el mismo código para experimentar también con los nuevos parámetros estudiados a continuación.

La sombra que obtuvimos hasta el momento es sólida, sin gradientes o transparencias (no realmente como una sombra suele aparecer). Existen algunos parámetros más y cambios que podemos implementar para mejorar la apariencia de la sombra.

Un cuarto valor que se puede agregar a la propiedad ya estudiada es la distancia de difuminación. Con este efecto ahora la sombra lucirá real. Puede intentar utilizar este nuevo parámetro declarando un valor de 10 pixeles a la regla del Listado 3-6, como en el siguiente ejemplo:

```
box-shadow: rgb(150,150,150) 5px 5px 10px;
```

Listado 3-7. *Agregando el valor de difuminación a* `box-shadow`*.*

Agregando otro valor más en pixeles al final de la propiedad desparramará la sombra. Este efecto cambia un poco la naturaleza de la sombra expandiendo el área que cubre. A pesar de que no recomendamos utilizar este efecto, puede ser aplicable en algunos diseños.

> **Hágalo usted mismo:** Intente agregar un valor de 20 pixeles al final del estilo del Listado 3-7 y combine este código con el código del Listado 3-6 para probarlo.
>
> **IMPORTANTE:** Siempre recuerde que en este momento las propiedades estudiadas son experimentales. Para usarlas, debe declarar cada una agregando los prefijos correspondientes, como **-moz-** o **-webkit-**, de acuerdo al navegador que

usa (en este ejemplo, Firefox o Google Chrome).

El último valor posible para **box-shadow** no es un número sino más bien una palabra clave: **inset**. Esta palabra clave convierte a la sombra externa en una sombra interna, lo cual provee un efecto de profundidad al elemento afectado.

```
box-shadow: rgb(150,150,150) 5px 5px 10px inset;
```

Listado 3-8. *Sombra interna.*

El estilo en el Listado 3-8 mostrará una sombra interna alejada del borde de la caja por unos 5 pixeles y con un efecto de difuminación de 10 pixeles.

> **Hágalo usted mismo:** Los estilos de los Listados 3-7 y 3-8 son solo ejemplos. Para comprobar los efectos en su navegador debe aplicar estos cambios al grupo completo de reglas presentado en el Listado 3-6.
>
> **IMPORTANTE:** Las sombras no expanden el elemento o incrementan su tamaño, por lo que tendrá que controlar cuidadosamente que el espacio disponible es suficiente para que la sombra sea expuesta ycorrectamente dibujada en la pantalla.

Text-shadow

Ahora que conoce todo acerca de sombras probablemente estará pensando en generar una para cada elemento de su documento. La propiedad **box-shadow** fue diseñada especialmente para ser aplicada en cajas. Si intenta aplicar este efecto a un elemento ****, por ejemplo, la caja invisible ocupada por este elemento en la pantalla tendrá una sombra, pero no el contenido del elemento. Para crear sombras para figuras irregulares como textos, existe una propiedad especial llamada **text-shadow**:

```
body {
  text-align: center;
}
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
  border-radius: 20px;
  -moz-box-shadow: rgb(150,150,150) 5px 5px 10px;
  -webkit-box-shadow: rgb(150,150,150) 5px 5px 10px;
  box-shadow: rgb(150,150,150) 5px 5px 10px;
}
#titulo {
  font: bold 36px verdana, sans-serif;
  text-shadow: rgb(0,0,150) 3px 3px 5px;
}
```

Listado 3-9. *Generando una sombra para el título.*

Los valores para **text-shadow** son similares a los usados para **box-shadow**. Podemos declarar el color de la sombra, la distancia horizontal yvertical de la sombra con respecto al objeto y el radio de difuminación.

En el Listado 3-9 una sombra azul fue aplicada al título de nuestra plantilla con una distancia de 3 pixeles y un radio de difuminación de 5.

@font-face

Obtener un texto con sombra es realmente un muy buen truco de diseño, imposible de lograr con métodos previos, pero más que cambiar el texto en sí mismo solo provee un efecto tridimensional. Una sombra, en este caso, es como pintar un viejo coche, al final será el mismo coche. En este caso, será el mismo tipo de letra.

El problema con las fuentes o tipos de letra es tan viejo como la web. Usuarios regulares de la web a menudo tienen un número limitado de fuentes instaladas en sus ordenadores, usualmente estas fuentes son diferentes de un usuario a otro, y la mayoría de las veces muchos usuarios tendrán fuentes que otros no. Por años, los sitios webs solo pudieron utilizar un limitado grupo de fuentes confiables (un grupo básico que prácticamente todos los usuarios tienen instalados) y así presentar la información en pantalla.

La propiedad **`@font-face`** permite a los diseñadores proveer un archivo conteniendo una fuente específica para mostrar sus textos en la página. Ahora podemos incluir cualquier fuente que necesitemos con solo proveer el archivo adecuado:

```
body {
  text-align: center;
}
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
  border-radius: 20px;

  -moz-box-shadow: rgb(150,150,150) 5px 5px 10px;
  -webkit-box-shadow: rgb(150,150,150) 5px 5px 10px;
  box-shadow: rgb(150,150,150) 5px 5px 10px;
}
#titulo {
  font: bold 36px MiNuevaFuente, verdana, sans-serif;
  text-shadow: rgb(0,0,150) 3px 3px 5px;
}
@font-face {
  font-family: 'MiNuevaFuente';
  src: url('font.ttf');
}
```

Listado 3-10. *Nueva fuente para el título.*

Hágalo usted mismo: Descargue el archivo **`font.ttf`** desde nuestro sitio web o use uno que ya posea y cópielo en el mismo directorio (carpeta) de su archivo CSS. Para descargar el archivo, visite el siguiente enlace: *www.minkbooks.com/content/font.ttf*. Puede obtener más fuentes similares de forma gratuita en *www.moorstation.org/typoasis/designers/steffmann/.*

IMPORTANTE: El archivo conteniendo la fuente debe encontrarse en el mismo dominio que la página web (o en el mismo ordenador, en este caso). Esta es una restricción de algunos navegadores como Firefox, por ejemplo.

La propiedad **`@font-face`** necesita al menos dos estilos para declarar la fuente y cargar el archivo. El estilo construido con la propiedad **`font-family`** especifica el nombre que queremos otorgar a esta fuente en particular, y la propiedad **`src`** indica la URL del archivo con el código correspondiente a esa fuente. En el Listado 3-10, el nombre **`MiNuevaFuente`** fue asignado a nuestro nuevo tipo de letra y el archivo **`font.ttf`** fue indicado como el archivo correspondiente a esta fuente.

Una vez que la fuente es cargada, podemos comenzar a usarla en cualquier elemento del documento simplemente escribiendo su nombre (**`MiNuevaFuente`**). En el estilo **`font`** en la regla del Listado 3-10, especificamos que el título será mostrado con la nueva fuente o las alternativas **`verdana`** y **`sans-serif`** en caso de que la fuente incorporada no sea cargada apropiadamente.

Gradiente lineal

Los gradientes son uno de los efectos más atractivos entre aquellos incorporados en CSS3. Este efecto era prácticamente imposible de implementar usando técnicas anteriores pero ahora es realmente fácil de hacer usando CSS. Una propiedad **background** con algunos pocos parámetros es suficiente para convertir su documento en una página web con aspecto profesional:

```
body {
  text-align: center;
}
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
  border-radius: 20px;

  -moz-box-shadow: rgb(150,150,150) 5px 5px 10px;
  -webkit-box-shadow: rgb(150,150,150) 5px 5px 10px;
  box-shadow: rgb(150,150,150) 5px 5px 10px;

  background: -webkit-linear-gradient(top, #FFFFFF, #006699);
  background: -moz-linear-gradient(top, #FFFFFF, #006699);
}
#titulo {
  font: bold 36px MiNuevaFuente, verdana, sans-serif;
  text-shadow: rgb(0,0,150) 3px 3px 5px;
}
@font-face {
  font-family: 'MiNuevaFuente';
  src: url('font.ttf');
}
```

Listado 3-11. *Agregando un hermoso gradiente de fondo a nuestra caja.*

Los gradientes son configurados como fondos, por lo que podemos usar las propiedades **background** o **background image** para declararlos. La sintaxis para los valores declarados en estas propiedades es **linear-gradient(posición inicio, color inicial, color final)**. Los atributos de la función **linear-gradient()** indican el punto de comienzo y los colores usados para crear el gradiente. El primer valor puede ser especificado en pixeles, porcentaje o usando las palabras clave **top**, **bottom**, **left** y **right** (como hicimos en nuestro ejemplo). El punto de comienzo puede ser reemplazado por un ángulo para declarar una dirección específica del gradiente:

```
background: linear-gradient(30deg, #FFFFFF, #006699);
```

Listado 3-12. *Gradiente con un ángulo de dirección de 30 grados.*

También podemos declarar los puntos de terminación para cada color:

```
background: linear-gradient(top, #FFFFFF 50%, #006699 90%);
```

Listado 3-13. *Declarando puntos de terminación.*

Gradiente radial

La sintaxis estándar para los gradientes radiales solo difiere en unos pocos aspectos con respecto a la anterior. Debemos usar la función **radial-gradient()** y un nuevo atributo para la forma:

```
background: radial-gradient(center, circle, #FFFFFF 0%, #006699 200%);
```

Listado 3-14. *Gradiente radial.*

La posición de comienzo es el origen y puede ser declarada en pixeles, porcentaje o una combinación de las palabras clave **center**, **top**, **bottom**, **left** y **right**. Existen dos posibles valores para la forma (**circle** y **ellipse**) y la terminación para el color indica el color yla posición donde las transiciones comienzan.

Hágalo usted mismo: Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-14 para probar el efecto en su navegador (no olvide agregar los prefijos **-moz-** o **-webkit-** dependiendo del navegador que esté usando).

IMPORTANTE: En este momento el efecto de gradientes ha sido implementado por los navegadores en diferentes formas. Lo que hemos aprendido en este capítulo es el estándar propuesto por W3C (World Wide Web Consortium). Navegadores como Firefox y Google Chrome ya incorporan una implementación que trabaja con este estándar, pero Internet Explorer y otros aún se encuentran ocupados en ello. Como siempre, pruebe sus códigos en cada navegador disponible en el mercado para comprobar el estado actual de las diferentes implementaciones.

RGBA

Hasta este momento los colores fueron declarados como sólidos utilizando valores hexadecimales o la función **rgb()** para decimales. CSS3 ha agregado una nueva función llamada **rgba()** que simplifica la asignación de colores y transparencias. Esta función además resuelve un problema previo provocado por la propiedad **opacity**.

La función **rgba()** tiene cuatro atributos. Los primeros tres son similares a los usados en **rgb()** y simplemente declaran los valores para los colores rojo, verde y azul en números decimales del 0 al 255. El último, en cambio, corresponde a la nueva capacidad de opacidad. Este valor se debe encontrar dentro de un rango que va de 0 a 1, con 0 como totalmente transparente y 1 como totalmente opaco.

```
#titulo {
  font: bold 36px MiNuevaFuente, verdana, sans-serif;
  text-shadow: rgba(0,0,0,0.5) 3px 3px 5px;
}
```

Listado 3-15. *Mejorando la sombra del texto con transparencia.*

El Listado 3-15 ofrece un simple ejemplo que demuestra cómo los efectos son mejorados aplicando transparencia. Reemplazamos la función **rgb()** por **rgba()** en la sombra del título y agregamos un valor de opacidad/transparencia de **0.5**. Ahora la sombra de nuestro título se mezclará con el fondo, creando un efecto mucho más natural.

En previas versiones de CSS teníamos que usar diferentes técnicas en diferentes navegadores para hacer un elemento transparente. Todas presentaban el mismo problema: el valor de opacidad de un elemento era heredado por sus hijos. Ese problema fue resuelto por **rgba()** y ahora podemos asignar un valor de opacidad al fondo de una caja sin afectar su contenido.

Hágalo usted mismo: Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-15 para probar el efecto en su navegador.

HSLA

Del mismo modo que la función **rgba()** agrega un valor de opacidad a **rgb()**, la función **hsla()** hace lo mismo para la función **hsl()**.

La función **hsla()** es simplemente un función diferente para generar colores, pero es más intuitiva que **rgba()**. Algunos diseñadores encontrarán más fácil generar un set de colores personalizado utilizando **hsla()**. La sintaxis de esta función es:

hsla(tono, saturación, luminosidad, opacidad).

```
#titulo {
  font: bold 36px MiNuevaFuente, verdana, sans-serif;
  text-shadow: rgba(0,0,0,0.5) 3px 3px 5px;
  color: hsla(120, 100%, 50%, 0.5);
}
```

Listado 3-16. *Nuevo color para el título usando* `hsla()`.

Siguiendo la sintaxis, **tono** representa el color extraído de una rueda imaginaria y es expresado en grados desde 0 a 360. Cerca de 0 y 360 están los colores rojos, cerca de 120 los verdes y cerca de 240 los azules. El valor **saturación** es representado en porcentaje, desde 0% (escala de grises) a 100% (todo color o completamente saturado). La **luminosidad** es también un valor en porcentaje desde 0% (completamente oscuro) a 100% (completamente iluminado). El valor 50% representa luminosidad normal o promedio. El último valor, así como en **rgba()**, representa la opacidad.

> **Hágalo usted mismo:** Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-16 para probar el efecto en su navegador.

Outline

La propiedad **outline** es una vieja propiedad CSS que ha sido expandida en CSS3 para incluir un valor de desplazamiento. Esta propiedad era usada para crear un segundo borde, y ahora ese borde puede ser mostrado alejado del borde real del elemento.

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  outline: 2px dashed #000099;
  outline-offset: 15px;
}
```

Listado 3-17. *Agregando un segundo borde a la cabecera.*

En el Listado 3-17 agregamos a los estilos originalmente aplicados a la caja de nuestra plantilla un segundo borde de 2 pixeles con un desplazamiento de 15 pixeles. La propiedad **outline** tiene similares características y usa los mismos parámetros que **border**. La propiedad **outline-offset** solo necesita un valor en pixeles.

> **Hágalo usted mismo:** Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-17 para probar el efecto en su navegador.

Border-image

Los posibles efectos logrados por las propiedades **border** y **outline** están limitados a líneas simples y solo algunas opciones de configuración. La nueva propiedad **border-image** fue incorporada para superar estas limitaciones y dejar en manos del diseñador la calidad yvariedad de bordes disponibles ofreciendo la alternativa de utilizar imágenes propias.

> **Hágalo usted mismo:** Vamos a utilizar una imagen PNG que incluye diamantes para probar esta propiedad. Siga el siguiente enlace para descargar el archivo **diamonds.png** desde nuestro sitio web y luego copie este archivo en el mismo directorio (carpeta) donde se encuentra su archivo CSS: *www.minkbooks.com/ content/diamonds.png*.

La propiedad **border-image** toma una imagen y la utiliza como patrón. De acuerdo a los valores otorgados, la imagen es cortada como un pastel, las partes obtenidas son luego ubicadas alrededor del objeto para construir el borde.

Figura 3-1. *Este es el patrón desde el cual vamos a construir nuestro borde. Cada pieza es de 29 pixeles de ancho, como indica la figura.*

Para hacer el trabajo, necesitamos especificar tres atributos: el nombre del archivo de la imagen, el tamaño de las piezas que queremos obtener del patrón y algunas palabras clave para declarar cómo las piezas serán distribuidas alrededor del objeto.

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 29px;
  -moz-border-image: url("diamonds.png") 29 stretch;
  -webkit-border-image: url("diamonds.png") 29 stretch;
  border-image: url("diamonds.png") 29 stretch;
}
```

Listado 3-18. *Un borde personalizado para la cabecera.*

Con las modificaciones realizadas en el Listado 3-18 estamos definiendo un borde de 29 pixeles para la caja de nuestra cabecera y luego cargando la imagen **`diamonds.png`** para construir ese borde. El valor 29 en la propiedad **`border-image`** declara el tamaño de las piezas y **`stretch`** es uno de los métodos disponibles para distribuir estas piezas alrededor de la caja.

Existen tres valores posibles para el último atributo. La palabra clave **`repeat`** repetirá las piezas tomadas de la imagen todas las veces que sea necesario para cubrir el lado del elemento. En este caso, el tamaño de las piezas es preservado y la imagen será cortada si no existe más espacio para ubicarla. La palabra clave **`round`** considerará qué tan largo es el lado a ser cubierto y ajustará el tamaño de las piezas para asegurarse que cubren todo el lado y ninguna pieza es cortada. Finalmente, la palabra clave **`stretch`** (usada en el Listado 3-18) estira solo una pieza para cubrir el lado completo.

En nuestro ejemplo utilizamos la propiedad **`border`** para definir el tamaño del borde, pero se puede también usar **`border-with`** para especificar diferentes tamaños para cada lado del elemento (la propiedad **`border-with`** usa cuatro parámetros, con una sintaxis similar a **`margin`** y **`padding`**). Lo mismo ocurre con el tamaño de cada pieza, hasta cuatro valores pueden ser declarados para obtener diferentes imágenes de diferentes tamaños desde el patrón.

> **Hágalo usted mismo:** Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-18 para probar el efecto en su navegador.

Transform y transition

Los elementos HTML, cuando son creados, son como bloques sólidos e inamovibles. Pueden ser movidos usando código Javascript o aprovechando librerías populares como jQuery (*www.jquery.com*), por ejemplo, pero no existía un procedimiento estándar para este propósito hasta que CSS3 presentó las propiedades **`transform`** y **`transition`**.

Ahora ya no tenemos que pensar en cómo hacerlo. En su lugar, solo tenemos que conocer cómo ajustar unos pocos parámetros y nuestro sitio web puede ser tan flexible y dinámico como lo imaginamos.

La propiedad **`transform`** puede operar cuatro transformaciones básicas en un elemento: **`scale`** (escalar), **`rotate`** (rotar), **`skew`** (inclinar) y **`translate`** (trasladar o mover). Veamos cómo funcionan:

Transform: scale

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-transform: scale(2);
  -webkit-transform: scale(2);
}
```

Listado 3-19. *Cambiando la escala de la caja de la cabecera.*

En el ejemplo del Listado 3-19 partimos de los estilos básicos utilizados para la cabecera generada en el Listado 3-2 y aplicamos transformación duplicando la escala del elemento. La función **`scale`** recibe dos parámetros: el valor **`X`** para la escala horizontal y el valor **`Y`** para la escala vertical. Si solo un valor es provisto el mismo valor es aplicado a ambos parámetros.

Números enteros y decimales pueden ser declarados para la escala. Esta escala es calculada por medio de una matriz. Los valores entre 0 y 1 reducirán el elemento, un valor de 1 mantendrá las proporciones originales y valores mayores que 1 aumentarán las dimensiones del elemento de manera incremental.

Un efecto atractivo puede ser logrado con esta función otorgando valores negativos:

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-transform: scale(1,-1);
  -webkit-transform: scale(1,-1);
}
```

Listado 3-20. *Creando una imagen espejo con* `scale`.

En el Listado 3-20, dos parámetros han sido declarados para cambiar la escala de la caja **`principal`**. El primer valor, 1, mantiene la proporción original para la dimensión horizontal de la caja. El segundo valor también mantiene la proporción original, pero invierte el elemento verticalmente para producir el efecto espejo.

Existen también otras dos funciones similares a **`scale`** pero restringidas a la dimensión horizontal o vertical: **`scaleX`** y **`scaleY`**. Estas funciones, por supuesto, utilizan un solo parámetro.

> **Hágalo usted mismo:** Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-19 o 3-20 para probar el efecto en su navegador.

Transform: rotate

La función **`rotate`** rota el elemento en la dirección de las agujas de un reloj. El valor debe ser especificado en grados usando la unidad "**`deg`**":

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
```

```
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-transform: rotate(30deg);
  -webkit-transform: rotate(30deg);
}
```

Listado 3-21. *Rotando la caja.*

Si un valor negativo es declarado, solo cambiará la dirección en la cual el elemento es rotado.

Hágalo usted mismo: Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-21 para probar el efecto en su navegador.

Transform: skew

Esta función cambia la simetría del elemento en grados y en ambas dimensiones.

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-transform: skew(20deg);
  -webkit-transform: skew(20deg);
}
```

Listado 3-22. *Inclinar horizontalmente.*

La función **`skew`** usa dos parámetros, pero a diferencia de otras funciones, cada parámetro de esta función solo afecta una dimensión (los parámetros actúan de forma independiente). En el Listado 3-22, realizamos una operación **`transform`** a la caja de la cabecera para inclinarla. Solo declaramos el primer parámetro, por lo que solo la dimensión horizontal de la caja será modificada. Si usáramos los dos parámetros, podríamos alterar ambas dimensiones del objeto. Como alternativa podemos utilizar funciones diferentes para cada una de ellas: **`skewX`** y **`skewY`**.

Hágalo usted mismo: Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-22 para probar el efecto en su navegador.

Transform: translate

Similar a las viejas propiedades **`top`** y **`left`**, la función **`translate`** mueve o desplaza el elemento en la pantalla a una nueva posición.

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-transform: translate(100px);
  -webkit-transform: translate(100px);
}
```

Listado 3-23. *Moviendo la caja de la cabecera hacia la derecha.*

La función **`translate`** considera la pantalla como una grilla de pixeles, con la posición original del elemento usada como un punto de referencia. La esquina superior izquierda del elemento es la posición `0,0`, por lo que valores negativos moverán al objeto hacia la izquierda o hacia arriba de la posición original, yvalores positivos lo harán hacia la derecha o hacia abajo.

En el Listado 3-23, movimos la caja de la cabecera hacia la derecha unos 100 pixeles desde su posición original. Dos valores pueden ser declarados en esta función si queremos mover el elemento horizontal y verticalmente, o podemos usar funciones independientes llamadas **`translateX`** y **`translateY`**.

> **Hágalo usted mismo:** Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-23 para probar el efecto en su navegador.

Transformando todo al mismo tiempo

Aveces podría resultar útil realizar sobre un elemento varias transformaciones al mismo tiempo. Para obtener una propiedad **`transform`** combinada, solo tenemos que separar cada función a aplicar con un espacio:

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-transform: translateY(100px) rotate(45deg) scaleX(0.3);
  -webkit-transform: translateY(100px) rotate(45deg) scaleX(0.3);
}
```

Listado 3-24. *Moviendo, escalando y rotando el elemento con solo una línea de código.*

Una de las cosas que debe recordar en este caso es que el orden es importante. Esto es debido a que algunas funciones mueven el punto original y el centro del objeto, cambiando de este modo los parámetros que el resto de las funciones utilizarán para operar.

> **Hágalo usted mismo:** Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-24 para probar el efecto en su navegador.

Transformaciones dinámicas

Lo que hemos aprendido hasta el momento en este capítulo cambiará la forma de la web, pero la mantendrá tan estática como siempre. Sin embargo, podemos aprovecharnos de la combinación de transformaciones y pseudo clases para convertir nuestra página en una aplicación dinámica:

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;
}
#principal:hover{
  -moz-transform: rotate(5deg);
  -webkit-transform: rotate(5deg);
}
```

Listado 3-25. *Respondiendo a la actividad del usuario.*

En el Listado 3-25, la regla original del Listado 3-2 para la caja de la cabecera fue conservada intacta, pero una nueva regla fue agregada para aplicar efectos de transformación usando la vieja pseudo clase **`:hover`**. El resultado obtenido es que cada vez que el puntero del ratón pasa sobre esta caja, la propiedad **`transform`** rota la caja en 5 grados, y cuando el puntero se aleja la caja vuelve a rotar de regreso a su posición original. Este efecto produce una animación básica pero útil con nada más que propiedades CSS.

> **Hágalo usted mismo:** Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-25 para probar el efecto en su navegador.

Transiciones

De ahora en más, hermosos efectos usando transformaciones dinámicas son accesibles y fáciles de implementar. Sin embargo, una animación real requiere de un proceso de más de dos pasos.

La propiedad **`transition`** fue incluida para suavizar los cambios, creando mágicamente el resto de los pasos que se encuentran implícitos en el movimiento. Solo agregando esta propiedad forzamos al navegador a tomar cartas en el asunto, crear para nosotros todos esos pasos invisibles, y generar una transición suave desde un estado al otro.

```
#principal {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-transition: -moz-transform 1s ease-in-out 0.5s;
  -webkit-transition: -webkit-transform 1s ease-in-out 0.5s;
}
#principal:hover{
  -moz-transform: rotate(5deg);
  -webkit-transform: rotate(5deg);
}
```

Listado 3-26. *Una hermosa rotación usando transiciones.*

Como puede ver en el Listado 3-26, la propiedad **`transition`** puede tomar hasta cuatro parámetros separados por un espacio. El primer valor es la propiedad que será considerada para hacer la transición (en nuestro ejemplo elegimos **`transform`**). Esto es necesario debido a que varias propiedades pueden cambiar al mismo tiempo y probablemente necesitemos crear los pasos del proceso de transición solo para una de ellas. El segundo parámetro especifica el tiempo que la transición se tomará para ir de la posición inicial a la final. El tercer parámetro puede ser cualquiera de las siguientes palabras clave: **`ease`**, **`linear`**, **`ease-in`**, **`ease-out`** o **`ease-in-out`**. Estas palabras clave determinan cómo se realizará el proceso de transición basado en una curva Bézier. Cada una de ellas representa diferentes tipos de curva Bézier, y la mejor forma de saber cómo trabajan es viéndolas funcionar en pantalla. El último parámetro para la propiedad **`transition`** es el retardo. Éste indica cuánto tiempo tardará la transición en comenzar.

Para producir una transición para todas las propiedades que están cambiando en un objeto, la palabra clave **`all`** debe ser especificada. También podemos declarar varias propiedades a la vezlistándolas separadas por coma.

> **Hágalo usted mismo:** Reemplace el correspondiente código del Listado 3-11 por el código del Listado 3-26 para probar el efecto en su navegador.
>
> **IMPORTANTE:** En el Listado 3-26 realizamos una transición con la propiedad **`transform`**. No todas las propiedades CSS son soportadas por la propiedad **`transition`** en este momento y probablemente la lista cambie con el tiempo. Deberá probar cada una de ellas por usted mismo o visitar el sitio web oficial de cada navegador para encontrar más información al respecto.