Lógica y Estructura de Datos

CSS3

Técnico Superior en Desarrollo de Software

Esc. Superior Nº 49 "Cap. Gral. J.J. Urquiza"

Año 2017

Ing. Álvaro Hergenreder

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 y funcionalidad, no solo organización estructural o definición de secciones. En este nuevo paradigma, HTML se presenta junto con CSS y Javascript como un único instrumento integrado.
La función de cada tecnología ya ha sido explicada en el apunte anterior, 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 y su 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. 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 apunte vamos a revisar estilos CSS y explicar algunas técnicas básicas para definir la estructura de un documento.

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>) será posicionado debajo del anterior.

Si creamos un documento HTML con la intención de reproducir un sitio web tradicional, incluyendo en el diseño barras horizontales y dos columnas en el medio, sería el siguiente:

```
28 <!DOCTYPE html>
29 <html lang="es">
30 <head>
31   <meta charset="iso-8859-1">
32   <meta name="description" content="Ejemplo de HTML5">
33   <meta name="keywords" content="HTML5, CSS3, JavaScript">
34   <title>Este texto es el título del documento</title>
35   <link rel="stylesheet" href="misestilos.css">
36 </head>
37 <body>
```

```
38    <header>
39      <h1>Este es el título principal del sitio web</h1>
40    </header>
41    <nav>
42      <ul>
43        <li>Principal</li>
44        <li>Fotos</li>
45        <li>Videos</li>
46        <li>Contacto</li>
47      </ul>
48    </nav>
49    <section>
50      <article>
51        <header>
52          <hgroup>
53            <h1>Título del mensaje uno</h1>
54            <h2>Subtítulo del mensaje uno</h2>
55          </hgroup>
56          <p>publicado 10-12-2014</p>
57        </header>
58        Este es el texto de mi primer mensaje
59        <figure>
60          <img src="imagenes-videos-sonidos/paisaje.jpg">
61          <figcaption>
62             Esta es la imagen del primer mensaje
63           </figcaption>
64        </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 15-12-2014</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; 2014-2015
    </footer>
  </body>
  </html>
```

Listado 2-0. *Ejemplo HTML*

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 anterior 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 1-1.

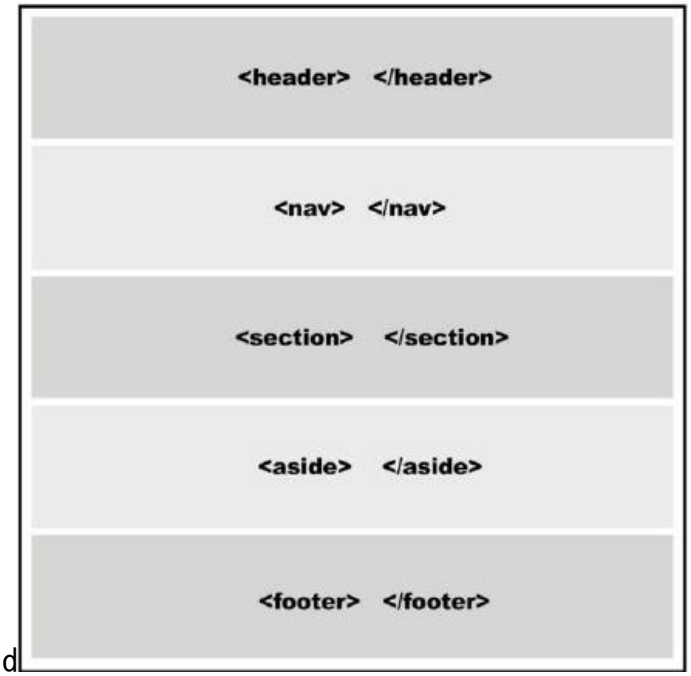

Figura 1-1. *Representación visual de una página web mostrada con estilos por defecto.*

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 apunte.
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: esto es solo una introducción breve a los estilos CSS. Solo mencionamos las técnicas y propiedades que necesita conocer para entender los temas.

Hágalo usted mismo: Dentro de un archivo de texto vacío, copie el código HTML del Listado 2-0 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.

En el apunte anterior de HTML, 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 y los 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 (o selector de atributo).
- Referencia por el atributo id (o selector id)
- Referencia por el atributo class (o selector de clase).

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 (etiqueta)

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 más adelante.

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 el color de fondo.

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, y son 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 y referenciamos 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 y estilos CSS fue posible reemplazar la función de tablas y efectivamente separar la estructura HTML de la presentación. Con elementos 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 y estilos CSS en un estándar. Finalmente el Modelo de Caja Tradicional fue adoptado a gran escala.

Plantilla

En el Listado 2-0 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 para agruparlas, centrarlas y darles un tamaño específico.
La nueva plantilla lucirá de este modo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf8">
  <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="2014-12-10" pubdate>publicado 10-12-2014</time>
      </header>
      Este es el texto de mi primer mensaje
      <figure>
        <img src="paisaje.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="2014-12-15" pubdate>publicado 15-12-2014</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; 2014-2015
  </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 2-0 mostrado al inicio del apunte. El primero es que ahora varias etiquetas fueron identificadas con los atributos id. Esto significa que podemos referenciar un elemento específico desde las reglas CSS con el valor de su atributo id.

El segundo cambio realizado a la vieja plantilla es la adición del elemento mencionado anteriormente. Este fue identificado con el atributo y el valor id=" agrupar" , y es cerrado al final del cuerpo con la etiqueta de cierre . 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 2-0 con el código en el Listado 2-25 y ubique las etiquetas de apertura y cierre del elemento 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.

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 y luego copiar en su interior la regla del Listado 2-26 y todas 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. 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 y tipo 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.*

A partir 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 un 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 id 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. A diferencia 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 y los 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. A sí 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 y las cajas se superponen.

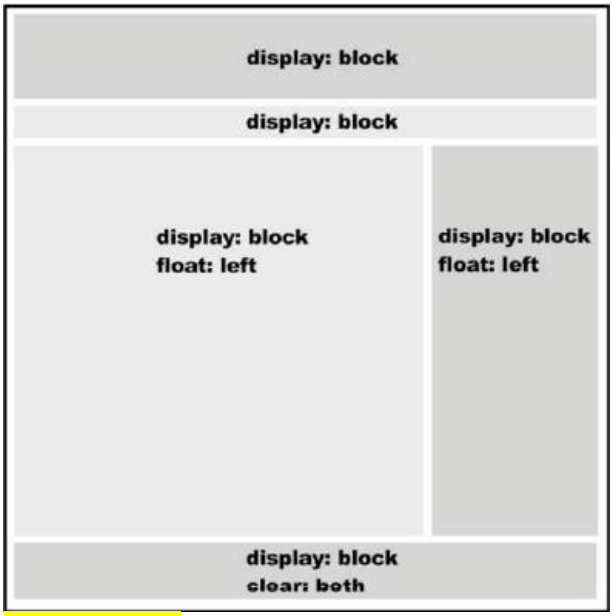

Figura 2-2. Representación visual del modelo de caja tradicional.

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, y luego 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.

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 y trabajar 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 que tiene otros elementos 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 .

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 .

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.

2.7 ¿Cómo definimos el color?

Tanto para el texto como para el fondo de diferentes elementos, entre otros, tenemos las siguientes alternativas para definir el color de los mismos.

- **Por palabras clave:**
  CSS define 17 palabras clave para referirse a los colores básicos. Las palabras se corresponden con el nombre en inglés de cada color:
  `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `orange`, `purple`, `red`, `silver`, `teal`, `white`, `yellow`.
  Por ejemplo: `p { color: orange }` estable un color naranja al párrafo.
  Aunque es una forma muy sencilla de referirse a los colores básicos, este método prácticamente no se utiliza en las hojas de estilos de los sitios web reales, ya que se trata de una gama de colores muy limitada.

- **Formato Decimal:**
  En el campo del diseño gráfico, se han definido varios modelos para hacer referencia a los colores. El modelo más conocido es RGB. Simplificando su explicación, el modelo RGB consiste en definir un color indicando la cantidad de color rojo, verde y azul que se debe *mezclar* para obtener ese color. Técnicamente, el modelo RGB es un modelo de tipo "aditivo", ya que los colores se obtienen sumando sus componentes.
  Por lo tanto, en el modelo RGB un color se define indicando sus tres componentes R (rojo), G (verde) y B (azul). Cada una de las componentes puede tomar un valor entre cero y un valor máximo. De esta forma, el color rojo puro en RGB se crea mediante el máximo valor de la componente R y un valor de 0 para las componentes G y B.
  Si todas las componentes valen 0, el color creado es el negro y si todas las componentes toman su valor máximo, el color obtenido es el blanco. En CSS, las componentes de los colores definidos mediante RGB decimal pueden tomar valores entre 0 y 255. El siguiente ejemplo establece el color del texto de un párrafo:
  `p { color: rgb(71,98,176) }`
  Si se indica un valor menor que 0 para una componente, automáticamente se transforma su valor en 0. Igualmente, si se indica un valor mayor que 255, se transforma automáticamente su valor a 255.

- **Formato Porcentual:**
  Las componentes RGB de un color también se pueden indicar mediante un porcentaje. El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal. La única diferencia es que en este caso el valor de las componentes RGB puede tomar valores entre 0% y 100%. Por tanto, para

transformar un valor RGB decimal en un valor RGB porcentual, es preciso realizar una regla de tres considerando que 0 es igual a 0% y 255 es igual a 100%.
El mismo color del ejemplo anterior se puede representar de forma porcentual:
p { color: rgb(27%,38%,69%)}
Al igual que sucede con el RGB decimal, si se indica un valor inferior a 0%, se transforma automáticamente en 0% y si se indica un valor superior a 100%, se trunca su valor a 100%.

- **Formato Hexadecimal:**
Aunque es el método más complicado para indicar los colores, se trata del método más utilizado con mucha diferencia. De hecho, prácticamente todos los sitios web reales utilizan exclusivamente este método (ver archivo *03-00-Los_colores-HTML-CSS.pdf* referido a la definición del color en formato hexadecimal).
Siguiendo el mismo ejemplo de las secciones anteriores, el color del párrafo se indica de la siguiente forma utilizando el formato RGB hexadecimal:
p { color: #4762B0 }
Recuerda que aunque es el método más complicado para definir un color, se trata del método que utilizan la inmensa mayoría de sitios web, por lo que es imprescindible dominarlo. Afortunadamente, todos los programas de diseño gráfico convierten de forma automática los valores RGB decimales a sus valores RGB hexadecimales, por lo que no tienes que hacer ninguna operación matemática.
El formato RGB hexadecimal es la forma más compacta de indicar un color, ya que incluso es posible comprimir sus valores cuando todas sus componentes son iguales dos a dos:
#AAA = #AAAAAA
#FFF = #FFFFFF
#A0F = #AA00FF
#369 = #336699
En el siguiente ejemplo se establece el color de fondo de la página a blanco, el color del texto a negro y el color de la letra de los titulares se define de color rojo:
body { background-color: #FFF; color: #000; }
h1, h2, h3, h4, h5, h6 { color: #C00; }
Las letras que forman parte del color en formato RGB hexadecimal se pueden escribir en mayúsculas o minúsculas indistintamente. No obstante, se recomienda escribirlas siempre en mayúsculas o siempre en minúsculas para que la hoja de estilos resultante sea más limpia y homogénea.

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 y construidos.

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 y satisfacer 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 y sombras hasta transformaciones y reposicionamiento 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 , <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 para líneas cortas y <p> para párrafos, entre otros. Por esta razón el texto en nuestra plantilla fue insertado usando etiquetas .

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 anterior). 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.*

Propiedades CSS3

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 y correctamente 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 y vertical 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.

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 y la 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 y variedad de bordes disponibles ofreciendo la alternativa de utilizar imágenes propias.

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.

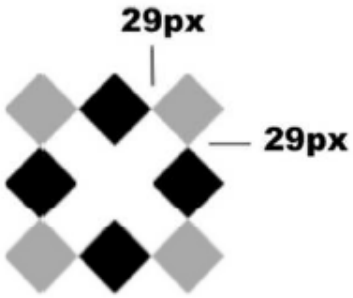

Figura 3-1. Este es el patrón desde el cual vamos a construir nuestro borde. Cada pieza es de 29 pixeles de ancho, como indica la figura.

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, y valores 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

A veces 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 vez listándolas 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.

3.2 Referencia rápida

CSS3 provee nuevas propiedades para crear efectos visuales y dinámicos que son parte esencial de la web en estos días.

border-radius Esta propiedad genera esquinas redondeadas para la caja formada por el elemento. Posee dos parámetros diferentes que dan forma a la esquina. El primer parámetro determina la curvatura horizontal y el segundo la vertical, otorgando la posibilidad de crear una elipsis. Para declarar ambos parámetros de la curva, los valores deben ser separados por una barra (por ejemplo, border-radius: 15px / 20px). Usando solo un valor determinaremos la misma forma para todas las esquinas (por ejemplo, border-radius: 20px). Un valor para cada esquina puede ser declarado en un orden que sigue las agujas del reloj, comenzando por la esquina superior izquierda.

box-shadow Esta propiedad crea sombras para la caja formada por el elemento. Puede tomar cinco parámetros: el color, el desplazamiento horizontal, el desplazamiento vertical, el valor de difuminación, y la palabra clave inset para generar una sombra interna. Los desplazamientos pueden ser negativos, y el valor de difuminación y el valor inset son opcionales (por ejemplo, box-shadow: #000000 5px 5px 10px inset).

text-shadow Esta propiedad es similar a box-shadow pero específica para textos. Toma cuatro parámetros: el color, el desplazamiento horizontal, el desplazamiento vertical, y el valor de difuminación (por ejemplo, text-shadow: #000000 5px 5px 10px).

@font-face Esta regla nos permite cargar y usar cualquier fuente que necesitemos. Primero, debemos declarar la fuente, proveer un nombre con la propiedad font-family y especificar el archivo con src (por ejemplo, @font-face{ font-family: Mifuente; src: url('font.ttf') }). Luego de esto, podremos asignar la fuente (en el ejemplo Mifuente) a cualquier elemento del documento.

linear-gradient*(posición inicio, color inicial, color final)* Esta función puede ser aplicada a las propiedades background o background-image para generar un gradiente lineal. Los atributos indican el punto inicial y los colores usados para crear el gradiente. El primer valor puede ser especificado en pixeles, en porcentaje o usando las palabras clave top, bottom, left y right. El punto de inicio puede ser reemplazado por un ángulo para proveer una dirección específica para el gradiente (por ejemplo, linear-gradient(top, #FFFFFF 50%, #006699 90%);).

radial-gradient*(posición inicio, forma, color inicial, color final)* Esta función puede ser aplicada a las propiedades background o background-image para generar un gradiente radial. La posición de inicio es el origen y puede ser declarado en

pixeles, porcentaje o como una combinación de las palabras clave center, top, bottom, left y right. Existen dos valores para la forma: circle y ellipse, y puntos de terminación pueden ser declarados para cada color indicando la posición donde la transición comienza (por ejemplo, radial-gradient(center, circle, #FFFFFF0%, #006699 200%);).

rgba() Esta función es una mejora de rgb(). Toma cuatro valores: el color rojo (0-255), el color verde (0-255), el color azul (0-255), y la opacidad (un valor entre 0 y 1).

hsla() Esta función es una mejora de hsl(). Puede tomar cuatro valores: el tono (un valor entre 0 y 360), la saturación (un porcentaje), la luminosidad (un porcentaje), y la opacidad (un valor entre 0 y 1).

outline Esta propiedad fue mejorada con la incorporación de otra propiedad llamada outline-offset. Ambas propiedades combinadas generan un segundo borde alejado del borde original del elemento (por ejemplo, outline: 1px solid #000000; outline-offset: 10px;).

border-image Esta propiedad crea un borde con una imagen personalizada. Necesita que el borde sea declarado previamente con las propiedades border o border-with, y toma al menos tres parámetros: la URL de la imagen, el tamaño de las piezas que serán tomadas de la imagen para construir el borde, y una palabra clave que especifica cómo esas piezas serán ubicadas alrededor del elemento (por ejemplo, border-image:url("file.png") 15 stretch;).

transform Esta propiedad modifica la forma de un elemento. Utiliza cuatro funciones básicas: scale (escalar), rotate (rotar), skew (inclinar), y translate (trasladar o mover). La función scale recibe solo un parámetro. Un valor negativo invierte el elemento, valores entre 0 y 1 reducen el elemento y valores mayores que 1 expanden el elemento (por ejemplo, transform: scale(1.5);). La función rotate usa solo un parámetro expresado en grados para rotar el elemento (por ejemplo, transform: rotate(20deg);). La función skew recibe dos valores, también en grados, para la transformación horizontal y vertical (por ejemplo, transform: skew(20deg, 20deg);). La función translate mueve el objeto tantos pixeles como sean especificados por sus parámetros (por ejemplo, transform: translate(20px);).