hello_

03.1

COLECCIÓN FRONTEND

CSS3 for Everybody

(porque el estilo no debería ser un dolor de cabeza)

Aprende desde cero cómo funciona CSS

Crea diseños flexibles, coloridos y responsivos

Domina propiedades, clases, layouts y

un día descubrí que display: flex; es magia negra… pero útil

@mihifidem

Desarrollador front-end, formador y luchador incansable contra los !important

CSS3

03.1

Tabla de contenidos

CSS3 -03.1

Introducción

1 Estilos **5**
Aplicando estilos 6
Hojas de estilo en cascada 9

2 Referencias **10**
Nombres 10
Atributo Id 14
Atributo Class 15
Otros atributos 17
Seudoclases 18

3 Propiedades **23**
Texto 23
Colores 29
Tamaño 31
Fondo 37
Bordes 41
Sombras 48
Gradientes 51
Filtros 56
Transformaciones 58
Transiciones 65
Animaciones 67

Listado de ejercicios 72

Introducción

CSS (Cascading Style Sheets) es un lenguaje de estilo utilizado para describir la presentación de un documento HTML o XML. CSS3 es la tercera versión de CSS y agrega nuevas funcionalidades y mejoras a la presentación web, incluyendo animaciones, transformaciones y estilos más complejos para los elementos HTML. CSS3 permite a los desarrolladores web crear diseños más atractivos y dinámicos, sin tener que recurrir a scripts externos.

CSS3 es una tecnología muy importante para la creación de sitios web modernos y atractivos. Algunas de las nuevas características de CSS3 incluyen:

- Selectores más avanzados: CSS3 permite a los desarrolladores web seleccionar elementos HTML de manera más precisa y eficiente.

- Mejoras en la disposición: CSS3 agrega nuevas propiedades para controlar la disposición de los elementos HTML, como el uso de columnas y el control de la visibilidad de elementos.

- Animaciones: CSS3 permite a los desarrolladores web crear animaciones con una sintaxis simple y eficiente.

- Estilos de bordes: CSS3 agrega nuevos estilos de bordes, como bordes redondeados y sombras, que pueden ser utilizados para crear diseños más atractivos y dinámicos.

- Propiedades de media: CSS3 agrega nuevas propiedades para controlar la presentación de los elementos HTML en diferentes tipos de dispositivos, como pantallas y impresoras.

CSS3 es compatible con los principales navegadores web y es una tecnología esencial para la creación de sitios web modernos y eficientes.

Tema 03.1
CSS

1. Estilos

En el capítulo anterior, hemos aprendido a crear un documento HTML, a organizar su estructura, y a determinar qué elementos son más apropiados para representar su contenido. Esta información nos permite definir el documento, pero no determina cómo se mostrará en pantalla. Desde la introducción de HTML5, esa tarea es responsabilidad de CSS. CSS es un lenguaje que facilita instrucciones que podemos usar para asignar estilos a los elementos HTML, como colores, tipos de letra, tamaños, etc. Los estilos se deben definir con CSS y luego asignar a los elementos hasta que logramos el diseño visual que queremos para nuestra página.

Por razones de compatibilidad, los navegadores asignan estilos por defecto a algunos elementos HTML. Esta es la razón por la que en el Capítulo 2 algunos elementos tenían márgenes o generaban saltos de línea, pero otros eran definidos de forma parecida (tenían el mismo tipo de letra y colores, por ejemplo). Algunos de estos estilos son útiles, pero la mayoría deben ser reemplazados o complementados con estilos personalizados. En CSS, los estilos personalizados se declaran con propiedades. Un estilo se define declarando el nombre de la propiedad y su valor separados por dos puntos. Por ejemplo, el siguiente código declara una propiedad que cambia el tamaño de la letra a 24 píxeles (debido a que algunas propiedades pueden incluir múltiples valores separados por un espacio, debemos indicar el final de la línea con un punto y coma).

```
font-size: 24px;
```

Listado 3-1: *Declarando propiedades CSS*

Si la propiedad del Listado 3-1 se aplica a un elemento, el texto contenido por ese elemento se mostrará en la pantalla con el tipo de letra definido por defecto, pero con un tamaño de 24 píxeles.

Lo básico: aunque los píxeles (**`px`**) son las unidades de medida que más se implementan, más adelante veremos que en algunas circunstancias, especialmente cuando creamos nuestro sitio web con diseño web adaptable, pueden ser más apropiadas otras unidades. Las unidades más utilizadas son **`px`** (píxeles), **`pt`** (puntos), **`in`** (pulgadas), **`cm`** (centímetros), **`mm`** (milímetros), **`em`** (relativo al tamaño de letra del elemento), **`rem`** (relativo al tamaño de letra del documento), y **`%`** (relativo al tamaño del contenedor del elemento).

La mayoría de las propiedades CSS pueden modificar un solo aspecto del elemento (el tamaño de la letra en este caso). Si queremos cambiar varios estilos al mismo tiempo, tenemos que declarar múltiples propiedades. CSS define una sintaxis que simplifica el proceso de asignar múltiples propiedades a un elemento. La construcción se llama regla. Una regla es una lista de propiedades declaradas entre llaves e identificadas por un selector. El selector indica qué elementos se verán afectados por las propiedades. Por ejemplo, la siguiente regla se identifica con el selector **p** y, por lo tanto, sus propiedades se aplicarán a todos los elementos **<p>** del documento.

```
p {
  color: #FF0000;
  font-size: 24px;
}
```

Listado 3-2: *Declarando reglas CSS*

La regla del Listado 3-2 incluye dos propiedades con sus respectivos valores agrupadas por llaves (color y font-size). Si aplicamos esta regla a nuestro documento, el texto dentro de cada elemento <p> se mostrará en color rojo y con un tamaño de 24 píxeles.

Aplicando estilos

Las propiedades y las reglas definen los estilos que queremos asignar a uno o más elementos, pero estos estilos no se aplican hasta que los incluimos en el documento. Existen tres técnicas disponibles para este propósito. Podemos usar estilos en línea, estilos incrustados u hojas de estilo. El primero, estilos en línea, utiliza un atributo global llamado style para insertar los estilos directamente en el elemento. Este atributo está disponible en cada uno de los elementos HTML y puede recibir una propiedad o una lista de propiedades que se aplicarán al elemento al que pertenece. Si queremos asignar estilos usando esta técnica, todo lo que tenemos que hacer es declarar el atributo style en el elemento que queremos modificar y asignarle las propiedades CSS.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
</head>
<body>
  <main>
    <section>
      <p style="font-size: 20px;">Mi Texto</p>
    </section>
  </main>
</body>
</html>
```

Listado 3-3: Estilos en Línea

El documento del Listado 3-3 incluye un elemento <p> con el atributo style y el valor font-size: 20px;. Cuando el navegador lee este atributo, le asigna un tamaño de 20 píxeles al texto dentro del elemento <p>.

Ejercicio 3.1 :cree un nuevo archivo HTML con el código del Listado 3-3 y abra el documento en su navegador. Debería ver el texto "Mi Texto" con letras en el tamaño definido por la propiedad **`font-size`**. Intente cambiar el valor de esta propiedad para ver cómo se presentan en pantalla los diferentes tamaños de letra.

Los estilos en línea son una manera práctica de probar estilos y ver cómo modifican los elementos, pero no se recomiendan para proyectos extensos. La razón es que el atributo style solo afecta al elemento en el que se ha declarado. Si queremos asignar el mismo estilo a otros elementos, tenemos que repetir el código en cada uno de ellos, lo cual incrementa innecesariamente el tamaño del documento, complicando su actualización y mantenimiento. Por ejemplo, si más adelante decidimos que en lugar de 20 píxeles el tamaño del texto en cada elemento <p> deber ser de 24 píxeles, tendríamos que cambiar cada estilo en cada uno de los elementos <p> del documento completo y en todos los documentos de nuestro sitio web.

Una alternativa es la de insertar las reglas CSS en la cabecera del documento usando selectores que determinan los elementos que se verán afectados. Para este propósito, HTML incluye el elemento <style>.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <style>
    p {
      font-size: 20px;
    }
  </style>
</head>
<body>
  <main>
    <section>
      <p>Mi texto</p>
    </section>
  </main>
</body>
</html>
```

Listado 3-4: Incluyendo estilos en la cabecera del documento

La propiedad declarada entre las etiquetas <style> en el documento del Listado 3-4 cumple la misma función que la declarada en el documento del Listado 3-3, pero en este ejemplo no tenemos que escribir el estilo dentro del elemento <p> que queremos modificar porque, debido al selector utilizado, todos se ven afectados por esta regla.

Declarar estilos en la cabecera del documento ahorra espacio y hace que el código sea más coherente y fácil de mantener, pero requiere que copiemos las mismas reglas en cada uno de los documentos de nuestro sitio web. Debido a que la mayoría de las páginas compartirán el mismo diseño, varias de estas reglas se duplicarán. La solución es mover los estilos a un archivo CSS y luego usar el elemento <link> para cargarlo desde cada uno de los documentos que lo requieran. Este tipo de archivos se denomina hojas de estilo, pero no son más que archivos de texto con la lista de reglas CSS que queremos asignar a los elementos del documento.

Como hemos visto en el Capítulo 2, el elemento <link> se usa para incorporar recursos externos al documento. Dependiendo del tipo de recurso que queremos cargar, tenemos que declarar diferentes atributos y valores. Para cargar hojas de estilo CSS, solo necesitamos los atributos rel y href. El atributo rel significa relación y especifica la relación entre el documento y el archivo que estamos incorporando, por lo que debemos declararlo con el valor stylesheet para comunicarle al navegador que el recurso es un archivo CSS con los estilos requeridos parapresentar la página. Por otro lado, el atributo **`href`** declara la URL del archivo a cargar.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <main>
    <section>
      <p>Mi texto</p>
    </section>
  </main>
</body>
</html>
```

Listado 3-5: *Aplicando estilos CSS desde un archivo externo*

El documento del Listado 3-5 carga los estilos CSS desde el archivo misestilos.css. En este archivo tenemos que declarar las reglas CSS que queremos aplicar al documento, tal como lo hemos hecho anteriormente dentro del elemento <style>. El siguiente es el código que debemos insertar en el archivo misestilos.css para producir el mismo efecto logrado en ejemplos anteriores.

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

Listado 3-6: *Definiendo estilos CSS en un archivo externo*

La práctica de incluir estilos CSS en un archivo aparte es ampliamente utilizada por diseñadores y se recomienda para sitos web diseñados con HTML5, no solo porque podemos definir una sola hoja de estilo y luego incluirla en todos los documentos con el elemento <link>, sino porque podemos reemplazar todos los estilos a la vez simplemente cargando un archivo diferente, lo cual nos permite probar diferentes diseños y adaptar el sitio web a las pantallas de todos los dispositivos disponibles, tal como veremos en el Capítulo 5.

Ejercicio 3.2 :cree un archivo HTML con el código del Listado 3-5 y un archivo llamado misestilos.css con el código del Listado 3-6. Abra el documento en su navegador. Debería ver el texto dentro del elemento **<p>** con un tamaño de 20 píxeles. Cambie el valor de la propiedad **font-size** en el archivo CSS y actualice la página para ver cómo se aplica el estilo al documento.

Hojas de estilo en cascada

Una característica importante del CSS es que los estilos se asignan en cascada (de ahí el nombre hojas de estilo en cascada o Cascading Style Sheets en inglés). Los elementos en los niveles bajos de la jerarquía heredan los estilos asignados a los elementos en los niveles más altos. Por ejemplo, si asignamos la regla del Listado 3-6 a los elementos <section> en lugar de los elementos <p>, como se muestra a continuación, el texto en el elemento <p> de nuestro documento se mostrará con un tamaño de 20 píxeles debido a que este elemento es hijo del elemento <section> y, por lo tanto, hereda sus estilos.

```
section {
  font-size: 20px;
}
```

Listado 3-7: *Heredando estilos*

Los estilos heredados de elementos en niveles superiores se pueden reemplazar por nuevos estilos definidos para los elementos en niveles inferiores de la jerarquía. Por ejemplo, podemos declarar una regla adicional para los elementos <p> que sobrescriba la propiedad font-size definida para el elemento <section> con un valor diferente.

```
section {
  font-size: 20px;
}
p {
  font-size: 36px;
}
```

Listado 3-8: *Sobrescribiendo estilos*

Ahora, el elemento <p> se nuestro documento se muestra con un tamaño de 36 píxeles porque el valor de la propiedad font-size asignada a los elementos <section> se modifica con la nueva regla asignada a los elementos <p>.

Figura 3-1: *Estilos en cascada*

Ejercicio 3.3 :reemplace los estilos en su archivo misestilos.css por elcódigo del Listado 3-8 y abra el documento del Listado 3-5 en su navegador. Debería ver el texto dentro del elemento **<p>** con un tamaño de 36 píxeles, tal como muestra la Figura 3-1.

2. Referencias

A veces es conveniente declarar todos los estilos en un archivo externo y luego cargar ese archivo desde cada documento que lo necesite, pero nos obliga a implementar diferentes mecanismos para establecer la relación entre las reglas CSS y los elementos dentro del documento que se verán afectados por las mismas.

Existen varios métodos para seleccionar los elementos que serán afectados por una regla CSS. En ejemplos anteriores hemos utilizado el nombre del elemento, pero también podemos usar los valores de los atributos id y class para referenciar un solo elemento o un grupo de elementos, e incluso combinarlos para construir selectores más específicos.

Nombres

Una regla declarada con el nombre del elemento como selector afecta a todos los elementos de ese tipo encontrados en el documento. En ejemplos anteriores usamos el nombre p para modificar elementos <p>, pero podemos cambiar este nombre para trabajar con cualquier elemento en el documento que deseemos. Si en su lugar declaramos el nombre span, por ejemplo, se modificarán todos los textos dentro de elementos .

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

Listado 3-9: *Referenciando elementos* `<span>` *por su nombre*

Si queremos asignar los mismos estilos a elementos con nombres diferentes, podemos declarar los nombres separados por una coma. En el siguiente ejemplo, la regla afecta a todos los elementos <p> y encontrados en el documento.

```
p, span {
  font-size: 20px;
}
```

Listado 3-10: *Declarando reglas con múltiples selectores*

También podemos referenciar solo elementos que se encuentran dentro de un elemento en particular listando los selectores separados por un espacio. Estos tipos de selectores se llaman selectores de descendencia porque afectan a elementos dentro de otros elementos, sin importar el lugar que ocupan en la jerarquía.

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

Listado 3-11: *Combinando selectores*

La regla en el Listado 3-11 afecta solo a los elementos <p> que se encuentran dentro de un elemento <main>, ya sea como contenido directo o insertados en otros elementos. Por ejemplo, el siguiente documento incluye un sección principal con una cabecera y una sección adicional. Ambos elementos incluyen elementos <p> para representar su contenido. Si aplicamos la regla del Listado 3-11 a este documento, el texto dentro de cada elemento <p> se mostrará con un tamaño de 20 píxeles porque todos son descendientes del elemento <main>.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="misestilos.css">
</head>

<body>
  <main>
    <header>
      <h1>Título</h1>
      <p>Esta es la introducción</p>
    </header>
    <section>
      <p>Frase 1</p>
      <p>Frase 2</p>
      <p>Frase 3</p>
      <p>Frase 4</p>
    </section>
  </main>
</body>
</html>
```

Listado 3-12: *Probando selectores*

La regla del Listado 3-11 solo afecta a elementos <p> que se encuentran dentro del elemento <main>. Si, por ejemplo, agregamos un elemento <footer> al final del documento del Listado 3-12, los elementos <p> dentro de este pie de página no se verán modificados. La Figura 3-2 muestra lo que vemos cuando abrimos este documento en el navegador.

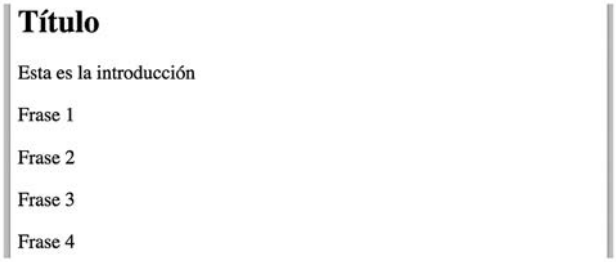

Figura 3-2: *Selectores de descendencia*

CSS incluye sintaxis adicionales para realizar una selección más precisa. Por ejemplo, podemos usar el carácter > para referenciar un elemento que es hijo directo de otro elemento.

```
section > p {
  font-size: 20px;
}
```

Listado 3-13: *Aplicando el selector* >

El carácter > indica que el elemento afectado es el elemento de la derecha cuando tiene como padre al elemento de la izquierda. La regla del Listado 3-13 modifica los elementos <p> que son hijos de un elemento **<section>**. Cuando aplicamos esta regla al documento del Listado 3-12, el elemento **<p>** dentro del elemento **<header>** no se ve afectado.

Figura 3-3: *Selector de hijo*

Con el carácter + se crea otro selector que facilita CSS. Este selector referencia un elemento que está precedido por otro elemento. Ambos deben compartir el mismo elemento padre.

```
h1 + p {
  font-size: 20px;
}
```

Listado 3-14: *Aplicando el selector +*

La regla del Listado 3-14 afecta a todos los elementos <p> que se ubican inmediatamente después de un elemento <h1>. Si aplicamos esta regla a nuestro documento, solo se modificará el elemento <p> dentro de la cabecera porque es el único que está precedido por un elemento <h1>.

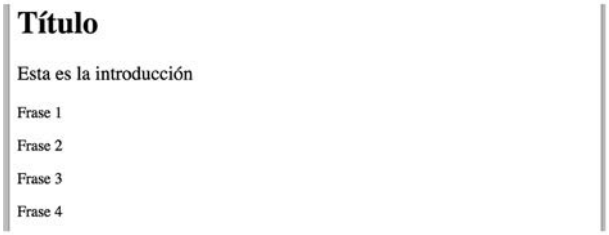

Figura 3-4: *Selector del elemento adyacente*

El selector anterior afecta solo al elemento <p> que se ubica inmediatamente después de un elemento <h1>. Si se coloca otro elemento entre ambos elementos, el elemento <p> no se modifica. CSS incluye el carácter ~ para crear un selector que afecta a todos los elementos que se ubican a continuación de otro elemento. Este selector es similar al anterior, pero el elemento afectado no tiene que encontrarse inmediatamente después del primer elemento. Esta regla afecta a todos los elementos encontrados, no solo al primero.

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

Listado 3-15: *Aplicando el selector ~*

La regla del Listado 3-15 afecta a todos los elementos <p> que preceden a otro elemento <p>. En nuestro ejemplo, se modificarán todos los elementos <p> dentro del elemento <section>, excepto el primero, porque no existe un elemento <p> que preceda al primer elemento <p>.

Figura 3-5: *Selector general de hermano*

Ejercicio 3.4 :cree un nuevo archivo HTML con el código del Listado 3-12. Cree un archivo CSS llamado misestilos.css y escriba dentro la regla que quiere probar. Inserte otros elementos entre los elementos **<p>** para ver cómo trabajan estas reglas.

Atributo Id

Las reglas anteriores afectan a los elementos del tipo indicado por el selector. Para seleccionar un elemento HTML sin considerar su tipo, podemos usar el atributo id. Este atributo es un nombre, un identificador exclusivo del elemento y, por lo tanto, lo podemos usar para encontrar un elemento en particular dentro del documento. Para referenciar un elemento usando su atributo id, el selector debe incluir el valor del atributo precedido por el carácter numeral (#).

```
#mitexto {
  font-size: 20px;
}
```

Listado 3-16: *Referenciando por medio del valor del atributo* `id`

La regla del Listado 3-16 solo se aplica al elemento identificado por el atributo id y el valor "mitexto", como el elemento <p> incluido en el siguiente documento.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <main>
    <section>
      <p>Frase 1</p>
      <p id="mitexto">Frase 2</p>
      <p>Frase 3</p>
      <p>Frase 4</p>
    </section>
  </main>
</body>
</html>
```

Listado 3-17: *Identificando un elemento* `<p>` *por medio de su atributo* `id`

La ventaja de este procedimiento es que cada vez que creamos una referencia usando el identificador mitexto en nuestro archivo CSS, solo se modifica el elemento con esa identificación, pero el resto de los elementos no se ven afectados. Esta es una forma muy específica de referenciar a un elemento y se usa comúnmente con elementos estructurales, como <section> o . Debido a su especificidad, el atributo id también se usa frecuentemente para referenciar elementos desde JavaScript, tal como veremos en los próximos capítulos.

Atributo Class

En lugar de usar el atributo id para asignar estilos, en la mayoría de las ocasiones es mejor hacerlo con el atributo class. Este atributo es más flexible y se puede asignar a varios elementos dentro del mismo documento.

```
.mitexto {
  font-size: 20px;
}
```

Listado 3-18: *Referenciando por medio del valor del atributo* `class`

Para referenciar un elemento usando su atributo class, el selector debe incluir el valor del atributo precedido por un punto. Por ejemplo, la regla del Listado 3-18 afecta a todos los elementos que contienen un atributo class con el valor "mitexto", como en el siguiente ejemplo.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="misestilos.css">
</head>

<body>
  <main>
    <section>
      <p class="mitexto">Frase 1</p>
      <p class="mitexto">Frase 2</p>
      <p>Frase 3</p>
      <p>Frase 4</p>
    </section>
  </main>
</body>
</html>
```

Listado 3-19: *Asignando estilos con el atributo* `class`

Los elementos <p> de las dos primeras líneas dentro de la sección principal del documento del Listado 3-19 incluyen el atributo class con el valor "mitexto". Debido a que se puede aplicar la misma regla a diferentes elementos del documento, estos dos primeros elementos son afectados por la regla del Listado 3-18. Por otro lado, los dos últimos elementos <p> no incluyen el atributo class y, por lo tanto, se mostrarán con los estilos por defecto.

Las reglas asignadas a través del atributo class se denominan clases. A un mismo elemento se le pueden asignar varias clases. Todo lo que tenemos que hacer es declarar los nombres de las clases separados por un espacio (por ejemplo, class="texto1 texto2"). Las clases también se pueden declarar como exclusivas para un tipo específico de elementos declarando el nombre del elemento antes del punto.

```
p.mitexto {
  font-size: 20px;
}
```

Listado 3-20: *Declarando una clase solo para elementos* <p>

En el Listado 3-20, hemos creado una regla que referencia la clase llamada mitexto, pero solo para los elementos <p>. Otros elementos que contengan el mismo valor en su atributo class no se verán afectados por esta regla.

Otros atributos

Aunque estos métodos de referencia cubren una amplia variedad de situaciones, a veces son insuficientes para encontrar el elemento exacto que queremos modificar. Para esas situaciones en las que los atributos id y class no son suficientes, CSS nos permite referenciar un elemento por medio de cualquier otro atributo que necesitemos. La sintaxis para definir esta clase de selectores incluye el nombre del elemento seguido del nombre del atributo en corchetes.

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

Listado 3-21: *Referenciando solo elementos* `<p>` *que tienen un atributo* `name`

La regla del Listado 3-21 modifica solo los elementos <p> que tienen un atributo llamado name. Para emular lo que hemos hecho con los atributos id y class, también podemos incluir el valor del atributo.

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

Listado 3-22: *Referenciando elementos* `<p>` *que tienen un atributo* `name` *con el valor* `mitexto`

CSS nos permite combinar el carácter = con otros caracteres para realizar una selección más específica. Los siguientes caracteres son los más utilizados.

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

Listado 3-23: *Aplicando selectores de atributo*

Las reglas del Listado 3-23 se han construido con selectores que incluyen los mismos atributos y valores, pero referencian diferentes elementos, como se describe a continuación.

- El selector con los caracteres **~=** referencia cualquier elemento **<p>** con un atributo **name** cuyo valor incluye la palabra "**mi**" (por ejemplo, "**mi texto**", "**mi coche**").
- El selector con los caracteres **^=** referencia cualquier elemento **<p>** con el atributo **name** cuyo valor comienza en "**mi**" (por ejemplo, "**mitexto**", "**micoche**").
- El selector con los caracteres **$=** referencia cualquier elemento **<p>** con un atributo **name** cuyo valor termina en "**mi**" (por ejemplo, "**textomi**", "**cochemi**").
- El selector con los caracteres ***=** referencia cualquier elemento **<p>** con un atributo **name** cuyo valor contiene la cadena de caracteres "**mi**" (en este caso, la cadena de caracteres podría también encontrarse en el medio del texto, como en "**textomicoche**").

En estos ejemplos, usamos el elemento <p>, el atributo name y un texto arbitrario como "mi", pero se puede aplicar la misma técnica a cualquier atributo y valor que necesitemos.

Seudoclases

Las seudoclases son herramientas especiales de CSS que nos permiten referenciar elementos HTML por medio de sus características, como sus posiciones en el código o sus condiciones actuales. Las siguientes son las que más se utilizan.

:nth-child(valor**)**—Esta seudoclase selecciona un elemento de una lista de elementos hermanos que se encuentra en la posición especificada por el valor entre paréntesis.

:first-child—Esta seudoclase selecciona el primer elemento de una lista de elementos hermanos.

:last-child—Esta seudoclase selecciona el último elemento de una lista de elementos hermanos.

:only-child—Esta seudoclase selecciona un elemento cuando es el único hijo de otro elemento.

:first-of-type—Esta seudoclase selecciona el primer elemento de una lista de elementos del mismo tipo.

:not(selector**)**—Esta seudoclase selecciona los elementos que no coinciden con el selector entre paréntesis.

Con estos selectores, podemos realizar una selección más dinámica. En los siguientes ejemplos, vamos a aplicar algunos utilizando un documento sencillo.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <main>
    <section>
      <p class="mitexto1">Frase 1</p>
      <p class="mitexto2">Frase 2</p>
      <p class="mitexto3">Frase 3</p>
      <p class="mitexto4">Frase 4</p>
    </section>
  </main>
</body>
</html>
```

Listado 3-24: *Creando un documento para probar las seudoclases*

La sección principal del documento del Listado 3-24 incluye cuatro elementos <p> que, considerando la estructura HTML, son hermanos y, por lo tanto, hijos del mismo elemento <section>. Usando seudoclases, podemos aprovechar esta organización y referenciar elementos sin importar cuánto conocemos acerca de sus atributos o valores. Por ejemplo, con la seudoclase :nth-child() podemos modificar todos los segundos elementos <p> que se encuentran en el documento.

```
p:nth-child(2) {
  font-size: 20px;
}
```

Listado 3-25: *Implementando la seudoclase* `:nth-child()`

Lo básico: las seudoclases se pueden agregar a cualquiera de los selectores que hemos mencionado con anterioridad. En la regla del Listado 3-25 hemos referenciado los elementos **`<p>`** usando el nombre **`p`**, pero esta regla también se podría haber escrito como **`.miclase:nth-child(2)`**, por ejemplo, para referenciar todo elemento que es hijo de otro elemento y que incluye el atributo **`class`** con el valor "miclase".

Lo que indica la seudoclase :nth-child() es algo como "el hijo en la posición…", por lo que el número entre paréntesis corresponde al número de posición del elemento hijo, o índice. Usando este tipo de referencias, podemos, por supuesto, seleccionar cualquier elemento hijo que queramos simplemente cambiando el número de índice. Por ejemplo, la siguiente regla afectará solo al último elemento <p> de nuestro documento.

```
p:nth-child(4) {
  font-size: 20px;
}
```

Listado 3-26: *Cambiando el índice para afectar a un elemento diferente*

Es posible asignar estilos a cada elemento creando una regla similar para cada uno de ellos.

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

Listado 3-27: *Generando una lista con la seudoclase* `:nth-child()`

En el Listado 3-27, hemos usado la seudoclase :nth-child() y la propiedad background-color para generar una lista de opciones que son claramente diferenciadas por el color de fondo. El valor #999999 define un gris oscuro y el valor #CCCCCC define un gris claro. Por lo tanto, el primer elemento tendrá un fondo oscuro, el segundo un fondo claro, y así sucesivamente.

Figura 3-6: *La seudoclase* `:nth-child()`

Se podrían agregar más opciones a la lista incorporando nuevos elementos <p> en el código HTML y nuevas reglas con la seudoclase :nth-child() en la hoja de estilo. Sin embargo, esta técnica nos obligaría a extender el código innecesariamente y sería imposible aplicarla en sitios web con contenido dinámico. Una alternativa es la que ofrecen las palabras clave odd y even disponibles para esta seudoclase.

```
p:nth-child(odd) {
  background-color: #999999;
}
p:nth-child(even) {
  background-color: #CCCCCC;
}
```

Listado 3-28: *Implementando las palabras clave* `odd` *y* `even`

La palabra clave odd para la seudoclase :nth-child() afecta a los elementos <p> que son hijos de otro elemento y tienen un índice impar, y la palabra clave even afecta a aquellos que tienen un índice par. Usando estas palabras clave, solo necesitamos dos reglas para configurar la lista completa, sin importar lo larga que sea. Incluso cuantas más opciones o filas se incorporen más adelante al documento, menos necesario será agregar otras reglas al archivo CSS. Los estilos se asignarán automáticamente a cada elemento de acuerdo con su posición.

Además de esta seudoclase existen otras que están relacionadas y también nos pueden ayudar a encontrar elementos específicos dentro de la jerarquía, como :first-child, :last-child y :only-child. La seudoclase :first-child referencia solo el primer elemento hijo, :last-child referencia solo el último elemento hijo, y :only-child afecta a un elemento cuando es el único elemento hijo. Estas seudoclases no requieren palabras clave o ningún parámetro adicional y se implementan según muestra el siguiente ejemplo.

```
p:last-child {
  font-size: 20px;
}
```

Listado 3-29: *Usando* `:last-child` *para modificar el último elemento* `<p>` *de la lista*

Otra seudoclase importante es :not(). Con esta seudoclase podemos seleccionar elementos que no coinciden con un selector. Por ejemplo, la siguiente regla asigna un margen de 0 píxeles a todos los elementos con nombres diferentes de p.

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

Listado 3-30: *Aplicando estilos a todos los elementos excepto* `<p>`

En lugar del nombre del elemento podemos usar cualquier otro selector que necesitemos. En el siguiente listado, se verán afectados todos los elementos dentro del elemento <section> excepto aquellos que contengan el atributo class con el valor "mitexto2".

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

Listado 3-31: *Definiendo una excepción por medio del atributo* `class`

Cuando aplicamos esta última regla al código HTML del Listado 3-24, el navegador asigna los estilos por defecto al elemento <p> con la clase mitexto2 y asigna un margen de 0 píxeles al resto (el elemento <p> con la clase mitexto2 es el que no se verá afectado por la regla). El resultado se ilustra en la Figura 3-7. Los elementos <p> primero, tercero y cuarto se presentan sin margen, pero el segundo elemento <p> se muestra con los márgenes superiores e inferiores por defecto.

3. Propiedades

Las propiedades son la pieza central de CSS. Todos los estilos que podemos aplicar a un elemento se definen por medio de propiedades. Ya hemos introducido algunas en los ejemplos anteriores, pero hay cientos de propiedades disponibles. Para simplificar su estudio, se pueden clasificar en dos tipos: propiedades de formato y propiedades de diseño.

Las propiedades de formato se encargan de dar forma a los elementos y su contenido, mientras que las de diseño están enfocadas a determinar el tamaño y la posición de los elementos en la pantalla. Así mismo, las propiedades de formato se pueden clasificar según el tipo de modificación que producen. Por ejemplo, algunas propiedades cambian el tipo de letra que se usa para mostrar el texto, otras generan un borde alrededor del elemento, asignan un color de fondo, etc. En este capítulo vamos a introducir las propiedades de formato siguiendo esta clasificación.

Texto

Desde CSS se pueden controlar varios aspectos del texto, como el tipo de letra que se usa para mostrar en pantalla, el espacio entre líneas, la alineación, etc. Las siguientes son las propiedades disponibles para definir el tipo de letra, tamaño, y estilo de un texto.

font-family—Esta propiedad declara el tipo de letra que se usa para mostrar el texto. Se pueden declarar múltiples valores separados por coma para ofrecer al navegador varias alternativas en caso de que algunos tipos de letra no se encuentren disponibles en el ordenador del usuario. Algunos de los valores estándar son **`Georgia`**, **`"Times New Roman"`**, **`Arial`**, **`Helvetica`**, **`"Arial Black"`**, **`Gadget`**, **`Tahoma`**, **`Geneva`**, **`Helvetica`**, **`Verdana`**, **`Geneva`**, **`Impact`**, y **`sans-serif`** (los nombres compuestos por más de una palabra se deben declarar entre comillas dobles).

font-size—Esta propiedad determina el tamaño de la letra. El valor puede ser declarado en píxeles (**`px`**), porcentaje (**`%`**), o usando cualquiera de las unidades disponibles en CSS como **`em`**, **`rem`**, **`pt`**, etc. El valor por defecto es normalmente **`16px`**.

font-weight—Esta propiedad determina si el texto se mostrará en negrita o no. Los valores disponibles son **`normal`** y **`bold`**, pero también podemos asignar los valores **`100`**, **`200`**, **`300`**, **`400`**, **`500`**, **`600`**, **`700`**, **`800`**, y **`900`** para determinar el grosor de la letra (solo disponibles para algunos tipos de letra).

font-style—Esta propiedad determina el estilo de la letra. Los valores disponibles son **`normal`**, **`italic`**, y **`oblique`**.

font—Esta propiedad nos permite declarar múltiples valores al mismo tiempo. Los valores deben declararse separados por un espacio y en un orden preciso. El estilo y el grosor se deben declarar antes que el tamaño, y el tipo de letra al final (por ejemplo, **`font: bold 24px Arial,sans-serif`**).

Estas propiedades se deben aplicar a cada elemento (o elemento padre) cuyo texto queremos modificar. Por ejemplo, el siguiente documento incluye una cabecera con un título y una sección con un párrafo. Como queremos asignar diferentes tipos de letra al título y al texto, tenemos que incluir el atributo id en cada elemento para poder identificarlos desde las reglas CSS.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <span id="titulo">Hojas de Estilo en Cascada</span>
  </header>
  <section>
    <p id="descripcion">Hojas de estilo en cascada (o CSS, siglas en
inglés de Cascading Style Sheets) es un lenguaje de diseño gráfico para
definir y crear la presentación de un documento estructurado escrito en
HTML.</p>
  </section>
  <footer>
    <a href="http://www.twooweb.com">www.twooweb.com</a>
  </footer>
</body>
</html>
```

Listado 3-32: *Probando propiedades de formato*

Las reglas deben incluir todas las propiedades requeridas para asignar los estilos que deseamos. Por ejemplo, si queremos asignar un tipo de letra y un nuevo tamaño al texto, tenemos que incluir las propiedades font-family y font-size.

```
#titulo {
  font-family: Verdana, sans-serif;
  font-size: 26px;
}
```

Listado 3-33: *Cambiando el tipo de letra y el tamaño del título*

Aunque podemos declarar varias propiedades en la misma regla para modificar diferentes aspectos del elemento, CSS ofrece una propiedad abreviada llamada font que nos permite definir todas las características del texto en una sola línea de código. Los valores asignados a esta propiedad se deben declarar separados por un espacio. Por ejemplo, la siguiente regla asigna el estilo bold (negrita), un tamaño de 26 píxeles, y el tipo de letra Verdana al elemento titulo.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-34: *Cambiando el tipo de letra con la propiedad* `font`

Cuando la regla del Listado 3-34 se aplica al documento del Listado 3-32, el título se muestra con los valores definidos por la propiedad font y el párrafo se presenta con los estilos por defecto.

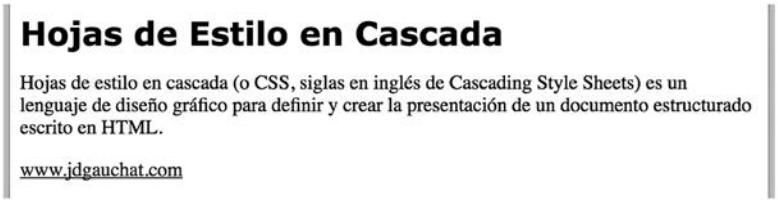

Figura 3-8: *La propiedad* `font`

Ejercicio 2.5 :cree un nuevo archivo HTML con el código del Listado 3-32. Cree o actualice el archivo misestilos.css con la regla del Listado 3-34. Abra el documento en su navegador. Debería ver algo parecido a lo que se muestra en la Figura 3-8.

Desde CSS podemos cambiar no solo el tipo de letra, sino también otros aspectos del texto, como el alineamiento, la sangría, el espacio entre líneas, etc. Las siguientes son algunas de las propiedades disponibles para este propósito.

text-align—Esta propiedad alinea el texto dentro de un elemento. Los valores disponibles son **`left`**, **`right`**, **`center`**, y **`justify`**.

text-align-last—Esta propiedad alinea la última línea de un párrafo. Los valores disponibles son **`left`**, **`right`**, **`center`**, y **`justify`**.

text-indent—Esta propiedad define el tamaño de la sangría de un párrafo (el espacio vacío al comienzo de la línea). El valor se puede declarar en píxeles (**`px`**), porcentaje (**`%`**), o usando cualquiera de las unidades disponibles en CSS, como **`em`**, **`rem`**, **`pt`**, etc.

letter-spacing—Esta propiedad define el espacio entre letras. El valor se debe declarar en píxeles (**`px`**), porcentaje (**`%`**) o usando cualquiera de las unidades disponibles en CSS, como **`em`**, **`rem`**, **`pt`**, etc.

word-spacing—Esta propiedad define el ancho del espacio entre palabras. El valor puede ser declarado en píxeles (**`px`**), porcentaje (**`%`**) o usando cualquiera de las unidades disponibles en CSS, como **`em`**, **`rem`**, **`pt`**, etc.

line-height—Esta propiedad define el espacio entre líneas. El valor se puede declarar en píxeles (**px**), porcentaje (**%**) o usando cualquiera de las unidades disponibles en CSS, como **em**, **rem**, **pt**, etc.

vertical-align—Esta propiedad alinea elementos verticalmente. Se usa frecuentemente para alinear texto con imágenes (la propiedad se aplica a la imagen). Los valores disponibles son **baseline**, **sub**, **super**, **text-top**, **text-bottom**, **middle**, **top**, y **bottom**.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  text-align: center;
}
```

Listado 3-35: *Alineando el texto con la propiedad* `text-align`

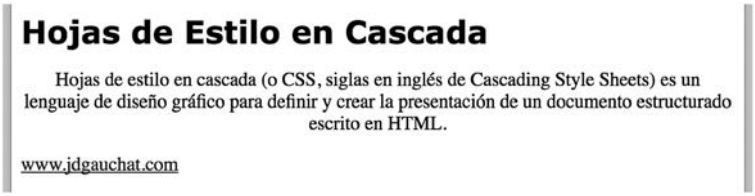

Figura 3-9: *La propiedad* `text-align`

El resto de las propiedades listadas arriba son simples de aplicar. Por ejemplo, podemos definir el tamaño del espacio entre palabras con la propiedad word-spacing.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  text-align: center;
  word-spacing: 20px;
}
```

Listado 3-36: *Definiendo el espacio entre palabras con la propiedad* `word-spacing`

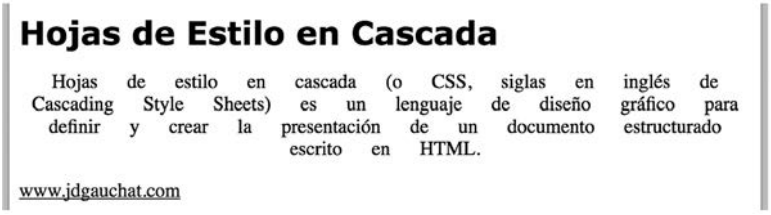

Figura 3-10: *La propiedad* `word-spacing`

CSS también ofrece la siguiente propiedad para decorar el texto con una línea.

text-decoration—Esta propiedad resalta el texto con una línea. Los valores disponibles son **underline**, **overline**, **line-through**, y **none**.

La propiedad text-decoration es particularmente útil con enlaces. Por defecto, los navegadores muestran los enlaces subrayados. Si queremos eliminar la línea, podemos declarar esta propiedad con el valor none. El siguiente ejemplo agrega una regla a nuestra hoja de estilo para modificar los elementos <a> del documento.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  text-align: center;
}
a {
  text-decoration: none;
}
```

Listado 3-37: *Eliminando las líneas en los enlaces de nuestro documento*

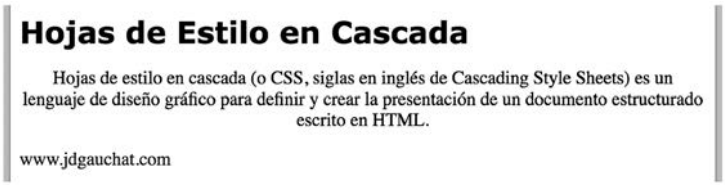

Figura 3-11: *La propiedad* `text-decoration`

Hasta el momento, hemos utilizado el tipo de letra Verdana (y la alternativa sans-serif en caso de que Verdana no esté disponible). Este tipo de letra es parte de un grupo conocido como fuentes seguras porque se encuentran disponibles en la mayoría de los ordenadores y, por lo tanto, podemos usarla con seguridad. El problema con las fuentes en la Web es que los navegadores no las descargan desde el servidor del sitio web, las cargan desde el ordenador del usuario, y los usuarios tienen diferentes tipos de letra instaladas en sus sistemas. Si usamos una fuente que el usuario no posee, el diseño de nuestro sitio web se verá diferente. Usando fuentes seguras nos aseguramos de que nuestro sitio web se verá de igual manera en cualquier navegador u ordenador, pero el problema es que los tipos de letras incluidos en este grupo son pocos. Para ofrecer una solución y permitir plena creatividad al diseñador, CSS incluye la regla @font-face. La regla @font-face es una regla reservada que permite a los diseñadores incluir un archivo con la fuente de letra a usar para mostrar el texto de una página web. Si la usamos, podemos incluir cualquier fuente que deseemos en nuestro sitio web con solo facilitar el archivo que la contiene.

La regla @font-face necesita al menos dos propiedades para declarar la fuente y cargar el archivo. La propiedad font-family especifica el nombre que queremos usar para referenciar este tipo de letra y la propiedad src indica la URL del archivo con las especificaciones de la fuente (la sintaxis de la propiedad src requiere el uso de la función url() para indicar la URL del archivo). En el siguiente ejemplo, el nombre MiNuevaLetra se asigna a nuestra fuente y el archivo font.ttf se indica como el archivo a cargar.

```
#titulo {
  font: bold 26px MiNuevaLetra, Verdana, sans-serif;
}
@font-face {
  font-family: "MiNuevaLetra";
  src: url("font.ttf");
}
```

Listado 3-38: *Cargando un tipo de letra personalizado para el título*

Una vez que se carga la fuente, podemos usarla en cualquier elemento del documento por medio de su nombre (MiNuevaLetra). En la propiedad font de la regla #titulo del Listado 3-38, especificamos que el título se mostrará con la nueva fuente o con las fuentes alternativas Verdana y sans-serif si nuestra fuente no se ha cargado.

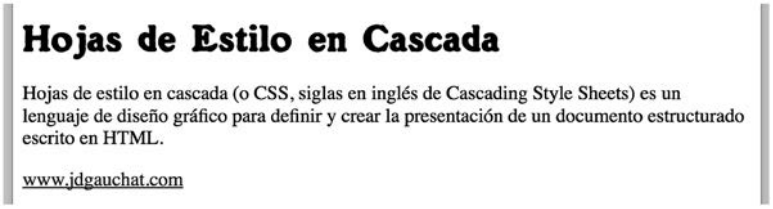

Figura 3-12: *Título con un tipo de letra personalizado*

Ejercicio 3.6 :descargue el archivo font.ttf desde nuestro sitio web. Copie el archivo dentro del directorio de su proyecto. Actualice su hoja de estilo con el código del Listado 3-38 y abra el documento del Listado 3-32 en su navegador. Puede encontrar más fuentes como la que se muestra en la Figura 3-12 en www.moorstation.org/typoasis/designers/steffmann/.

Lo básico: una función es un trozo de código que realiza una tarea específica y devuelve el resultado. La función **`url()`**, por ejemplo, tiene la tarea de cargar el archivo indicado entre paréntesis y devolver su contenido. CSS incluye varias funciones para generar los valores de sus propiedades. Introduciremos algunas de estas funciones a continuación y aprenderemos más sobre funciones en el Capítulo 6.

Colores

Existen dos formas de declarar un color en CSS: podemos usar una combinación de tres colores básicos (rojo, verde y azul), o definir el matiz, la saturación y la luminosidad. El color final se crea considerando los niveles que asignamos a cada componente. Dependiendo del tipo de sistema que utilizamos para definir el color, tendremos que declarar los niveles usando números hexadecimales (desde 00 a FF), números decimales (desde 0 a 255) o porcentajes. Por ejemplo, si decidimos usar una combinación de niveles de rojo, verde y azul, podemos declarar los niveles con números hexadecimales. En este caso, los valores del color se declaran en secuencia y precedidos por el carácter numeral, como en #996633 (99 es el nivel de rojo, 66 es el nivel de verde y 33 es el nivel de azul). Para definir colores con otros tipos de valores, CSS ofrece las siguientes funciones.

rgb(rojo, verde, azul**)**—Esta función define un color por medio de los valores especificados por los atributos (desde 0 a 255). El primer valor representa el nivel de rojo, el segundo valor representa el nivel de verde y el último valor el nivel de azul (por ejemplo, `rgb(153, 102, 51)`).

rgba(rojo, verde, azul, alfa**)**—Esta función es similar a la función `rgb()`, pero incluye un componente adicional para definir la opacidad (alfa). El valor se puede declarar entre 0 y 1, con 0 como totalmente transparente y 1 como totalmente opaco.

hsl(matiz, saturación, luminosidad**)**—Esta función define un color desde los valores especificados por los atributos. Los valores se declaran en números decimales y porcentajes.

hsla(matiz, saturación, luminosidad, alfa**)**—Esta función es similar a la función `hsl()`, pero incluye un componente adicional para definir la opacidad (alfa). El valor se puede declarar entre 0 y 1, con 0 como totalmente transparente y 1 como totalmente opaco.

Como veremos más adelante, son varias las propiedades que requieren valores que definen colores, pero la siguiente es la que se utiliza con más frecuencia:

color—Esta propiedad declara el color del contenido del elemento.

La siguiente regla asigna un gris claro al título de nuestro documento usando números hexadecimales.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
  color: #CCCCCC;
}
```

Listado 3-39: *Asignando un color al título*

Cuando los niveles de rojo, verde y azul son iguales, como en este caso, el color final se encuentra dentro de una escala de grises, desde negro (#000000) a blanco (#FFFFFF).

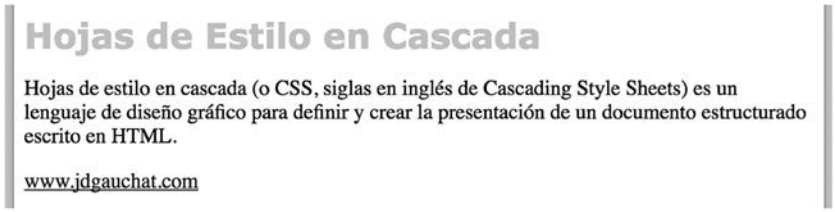

Figura 3-13: *Título con un color personalizado*

Declarar un color con una función es bastante parecido: solo tenemos que reemplazar el valor hexadecimal por la función que queremos utilizar. Por ejemplo, podemos definir el mismo color de grises con la función rgb() y valores decimales.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
  color: rgb(204, 204, 204);
}
```

Listado 3-40: *Asignando un color con la función* `rgb()`

La función hsl() es simplemente otra función disponible para generar un color, pero es más intuitiva que rgb(). A algunos diseñadores les resulta más fácil crear grupos de colores usando hsl(). Los valores requeridos por esta función definen el matiz, la saturación y la luminosidad. El matiz es un color extraído de una rueda imaginaria, expresado en grados desde 0 a 360: alrededor de 0 y 360 se encuentran los rojos, cerca de 120 los verdes, y cerca de 240 los azules. La saturación se representa en porcentaje, desde 0 % (escala de grises) a 100 % (todo color o totalmente saturado), y la luminosidad es también un valor en porcentaje, desde 0 % (completamente negro) a 100 % (completamente blanco); un valor de 50 % representa una luminosidad promedio. Por ejemplo, en la siguiente regla, la saturación se define como 0 % para crear un color dentro de la escala de grises y la luminosidad se especifica en 80 % para obtener un gris claro.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
  color: hsl(0, 0%, 80%);
}
```

Listado 3-41: *Asignando un color con la función* `hsl()`

IMPORTANTE: CSS define una propiedad llamada **`opacity`** para declarar la opacidad de un elemento. Esta propiedad presenta el problema de que el valor de opacidad para un elemento lo heredan sus elementos hijos. Ese problema se resuelve con las funciones **`rgba()`** y **`hsla()`**, con las que podemos asignar un valor de opacidad al fondo de un elemento, como veremos a continuación, sin que su contenido se vea afectado.

Lo básico: no es práctico encontrar los colores adecuados para nuestro sitio web combinando números y valores. Para facilitar esta tarea, los ordenadores personales incluyen varias herramientas visuales que nos permiten seleccionar un color y obtener su valor. Además, la mayoría de los editores de fotografía y programas gráficos disponibles en el mercado incluyen una herramienta que muestra una paleta desde donde podemos seleccionar un color y obtener el correspondiente valor hexadecimal o decimal. Para encontrar los colores adecuados, puede usar estas herramientas o cualquiera de las disponibles en la Web, como **www.colorhexa.com** o **htmlcolorcodes.com**.

Tamaño

Por defecto, el tamaño de la mayoría de los elementos se determina según el espacio disponible en el contenedor. El ancho de un elemento se define como 100 %, lo cual significa que será tan ancho como su contenedor, y tendrá una altura determinada por su contenido.

Esta es la razón por la que el elemento <p> del documento del Listado 3-32 se extendía a los lados de la ventana del navegador y no era más alto de lo necesario para contener las líneas del párrafo. CSS define las siguientes propiedades para declarar un tamaño personalizado:

width—Esta propiedad declara el ancho de un elemento. El valor se puede especificar en píxeles, porcentaje, o con la palabra clave **`auto`** (por defecto). Cuando el valor se especifica en porcentaje, el ancho se calcula según el navegador a partir del ancho del contenedor, y cuando se declara con el valor **`auto`**, el elemento se expande hasta ocupar todo el espacio horizontal disponible dentro del contenedor.

height—Esta propiedad declara la altura de un elemento. El valor se puede especificar en píxeles, porcentaje, o con la palabra clave **`auto`** (por defecto). Cuando el valor se especifica en porcentaje, el navegador calcula la altura a partir de la altura del contenedor, y cuando se declara con el valor **`auto`**, el elemento adopta la altura de su contenedor.

Los navegadores generan una caja alrededor de cada elemento que determina el área que ocupa en la pantalla. Cuando declaramos un tamaño personalizado, la caja se modifica y el contenido del elemento se adapta para encajar dentro de la nueva área, tal como muestra la Figura 3-14.

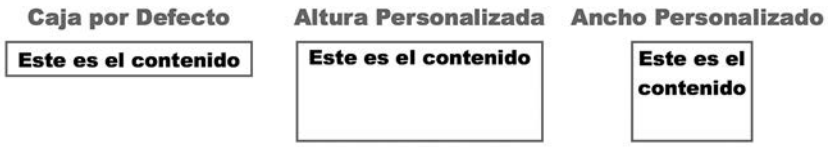

Figura 3-14: *Caja personalizada*

Por ejemplo, si declaramos un ancho de 200 píxeles para el elemento <p> de nuestro documento, las líneas del texto serán menos largas, pero se agregarán nuevas líneas para mostrar todo el párrafo.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  width: 200px;
}
```

Listado 3-42: *Asignando un ancho personalizado*

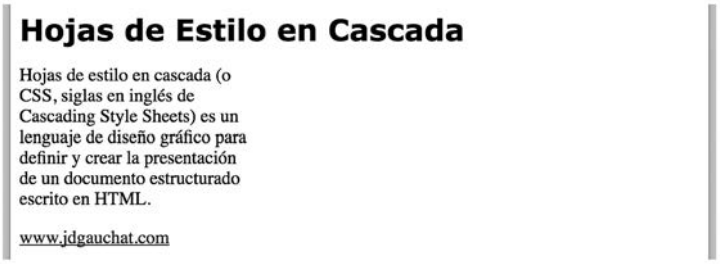

Figura 3-15: *Contenido principal con un tamaño personalizado*

La regla del Listado 3-42 declara el ancho del elemento <p>, pero la altura queda aún determinada por su contenido, lo que significa que la caja generada por este elemento será más alta para contener el párrafo completo, según ilustra la Figura 3-15. Si también queremos restringir la altura del elemento, podemos usar la propiedad height. La siguiente regla reduce la altura del elemento <p> de nuestro documento a 100 píxeles.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  width: 200px;
  height: 100px;
}
```

Listado 3-43: *Asignando una altura personalizada*

El problema con elementos que tienen un tamaño definido es que a veces el contenido no se puede mostrar en su totalidad. Por defecto, los navegadores muestran el resto del contenido fuera del área de la caja. En consecuencia, parte del contenido de una caja con tamaño personalizado se puede posicionar sobre el contenido del elemento que se encuentra debajo, tal como se ilustra en la Figura 3-16.

Figura 3-16: *Contenido desbordado*

En nuestro ejemplo, el texto que se encuentra fuera de la caja determinada por el elemento <p> ocupa el espacio correspondiente al elemento <footer> y, por lo tanto, el contenido de ambos elementos se encuentra superpuesto. CSS incluye las siguientes propiedades para resolver este problema:

overflow—Esta propiedad especifica cómo se mostrará el contenido que desborda el elemento. Los valores disponibles son **`visible`** (por defecto), **`hidden`** (esconde el contenido que no entra dentro de la caja), **`scroll`** (muestra barras laterales para desplazar el contenido), **`auto`** (deja que el navegador decida qué hacer con el contenido).

overflow-x—Esta propiedad especifica cómo se mostrará el contenido que desborda el elemento horizontalmente. Acepta los mismos valores que la propiedad **`overflow`**.

overflow-y—Esta propiedad especifica cómo se mostrará el contenido que desborda el elemento verticalmente. Acepta los mismos valores que la propiedad **`overflow`**.

overflow-wrap—Esta propiedad indica si un palabra debería ser dividida en un punto arbitrario cuando no hay suficiente espacio para mostrarla en la línea. Los valores disponibles son **`normal`** (la línea será dividida naturalmente) y **`break-word`** (las palabras se dividirán en puntos arbitrarios para acomodar la línea de texto en el espacio disponible).

Con estas propiedades podemos determinar cómo se mostrará el contenido cuando no hay suficiente espacio disponible. Por ejemplo, podemos ocultar el texto que desborda el elemento asignando el valor hidden a la propiedad overflow.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  width: 200px;
  height: 100px;
  overflow: hidden;
}
```

Listado 3-44: *Ocultando el contenido que desborda el elemento*

Figura 3-17: *Contenido ocultado*

Si queremos que el usuario pueda ver el texto que se ha ocultado, podemos asignar el valor scroll y forzar al navegador a mostrar barras laterales para desplazar el contenido.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  width: 200px;
  height: 100px;
  overflow: scroll;
}
```

Listado 3-45: *Incorporando barras de desplazamiento*

Figura 3-18: *Barras de desplazamiento*

El tamaño del elemento no queda solo determinado por el ancho y la altura de su caja, sino también por el relleno y los márgenes. CSS nos permite designar espacio alrededor de la caja para separar el elemento de otros elementos a su alrededor (margen), además de incluir espacio entre los límites de la caja y su contenido (relleno). La Figura 3-19 ilustra cómo se aplican estos espacios a un elemento.

Figura 3-19: *Márgenes, rellenos y bordes*

CSS incluye las siguientes propiedades para definir márgenes y rellenos para un elemento.

margin—Esta propiedad declara el margen de un elemento. El margen es el espacio que hay alrededor de la caja. Puede recibir cuatro valores que representan el margen superior, derecho, inferior, e izquierdo, en ese orden y separados por un espacio (por ejemplo, `margin: 10px 30px 10px 30px;`). Sin embargo, si solo se declaran uno, dos o tres valores, los otros toman los mismos valores (por ejemplo, `margin: 10px 30px` asigna 10 píxeles al margen superior e inferior y 30 píxeles al margen izquierdo y derecho). Los valores se pueden declarar independientemente usando las propiedades asociadas `margin-top`, `margin-right`, `margin-bottom` y `margin-left` (por ejemplo, `margin-left: 10px;`). La propiedad también acepta el valor `auto` para obligar al navegador a calcular el margen (usado para centrar un elemento dentro de su contenedor).

padding—Esta propiedad declara el relleno de un elemento. El relleno es el espacio entre el contenido del elemento y los límites de su caja. Los valores se declaran de la misma forma que lo hacemos para la propiedad `margin`, aunque también se pueden declarar de forma independiente con las propiedades `padding-top`, `padding-right`, `padding-bottom` y `padding-left` (por ejemplo, `padding-top: 10px;`).

La siguiente regla agrega márgenes y relleno a la cabecera de nuestro documento. Debido a que asignamos solo un valor, el mismo valor se usa para definir todos los márgenes y rellenos del elemento (superior, derecho, inferior e izquierdo, en ese orden).

```
header  {
  margin: 30px;
  padding: 15px;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-46: *Agregando márgenes y relleno*

El tamaño de un elemento y sus márgenes se agregan para calcular el área que ocupa. Lo mismo pasa con el relleno y el borde (estudiaremos bordes más adelante). El tamaño final de un elemento se calcula con la fórmula: tamaño + márgenes + relleno + bordes. Por ejemplo, si tenemos un elemento con un ancho de 200 píxeles y un margen de 10 píxeles a cada lado, el ancho del área ocupada por el elemento será de 220 píxeles. El total de 20 píxeles de margen se agrega a los 200 píxeles del elemento y el valor final se representa en la pantalla.

Figura 3-20: *Cabecera con márgenes y relleno personalizados*

Ejercicio 3.7 :reemplace las reglas en su archivo CSS con las reglas del Listado 3-46 y abra el documento en su navegador. Debería ver algo similar a lo que se representa en la Figura 3-20. Cambie el valor de la propiedad **`margin`** para ver cómo afecta a los márgenes.

IMPORTANTE: como veremos más adelante en este capítulo, los elementos se clasifican en dos tipos principales: Block (bloque) e Inline (en línea). Los elementos Block pueden tener un tamaño personalizado, pero los elementos Inline solo pueden ocupar el espacio determinado por sus contenidos. El elemento **`<span>`** que usamos para definir el título de la cabecera se declara por defecto como elemento Inline y, por lo tanto, no puede tener un tamaño y unos márgenes personalizados. Esta es la razón por la que en nuestro ejemplo asignamos márgenes y relleno al elemento **`<header>`** en lugar de al elemento **`<span>`** (los elementos estructurales se definen todos como elementos Block).

Fondo

Los elementos pueden incluir un fondo que se muestra detrás del contenido del elemento y a través del área ocupada por el contenido y el relleno. Debido a que el fondo puede estar compuesto por colores e imágenes, CSS define varias propiedades para generarlo.

background-color—Esta propiedad asigna un fondo de color a un elemento.

background-image—Esta propiedad asigna una o varias imágenes al fondo de un elemento. La URL del archivo se declara con la función **`url()`** (por ejemplo, **`url("ladrillos.jpg")`**). Si se requiere más de una imagen, los valores se deben separar por una coma.

background-position—Esta propiedad declara la posición de comienzo de una imagen de fondo. Los valores se pueden especificar en porcentaje, píxeles o usando una combinación de las palabras clave **`center`**, **`left`**, **`right`**, **`top`**, y **`bottom`**.

background-size—Esta propiedad declara el tamaño de la imagen de fondo. Los valores se pueden especificar en porcentaje, píxeles, o usando las palabras clave **`cover`** y **`contain`**. La palabra clave **`cover`** expande la imagen hasta que su ancho o su altura cubren el área del elemento, mientras que **`contain`** estira la imagen para ocupar toda el área del elemento.

background-repeat—Esta propiedad determina cómo se distribuye la imagen de fondo usando cuatro palabras clave: **`repeat`**, **`repeat-x`**, **`repeat-y`** y **`no-repeat`**. La palabra clave **`repeat`** repite la imagen en el eje vertical y horizontal, mientras que **`repeat-x`** y **`repeat-y`** lo hacen solo en el eje horizontal o vertical, respectivamente. Finalmente, **`no-repeat`** muestra la imagen de fondo una sola vez.

background-origin—Esta propiedad determina si la imagen de fondo se posicionará considerando el borde, el relleno o el contenido del área del elemento. Los valores disponibles son **`border-box`**, **`padding-box`**, y **`content-box`**.

background-clip—Esta propiedad declara el área a cubrir por el fondo usando los valores **`border-box`**, **`padding-box`**, y **`content-box`**. El primer valor corta la imagen al borde de la caja del elemento, el segundo corta la imagen en el relleno de la caja y el tercero corta la imagen alrededor del contenido de la caja.

background-attachment—Esta propiedad determina si la imagen es estática o se desplaza con el resto de los elementos usando dos valores: **`scroll`** (por defecto) y **`fixed`**. El valor **`scroll`** hace que la imagen se desplace con la página, y el valor **`fixed`** fija la imagen de fondo en su lugar original.

background—Esta propiedad nos permite declarar todos los atributos del fondo al mismo tiempo.

Los fondos más comunes se crean con colores. La siguiente regla implementa la propiedad background-color para agregar un fondo gris a la cabecera de nuestro documento.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  background-color: #CCCCCC;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-47: *Agregando un color de fondo*

Debido a que definimos márgenes de 30 píxeles a los lados, la cabecera queda centrada en la ventana del navegador, pero el texto dentro del elemento <header> aún se encuentra alineado a la izquierda (la alineación por defecto). En el ejemplo del Listado 3-47, además de cambiar el fondo, también incluimos la propiedad text-align para centrar el título. El resultado se muestra en la Figura 3-21.

Figura 3-21: *Fondo de color*

Además de colores, también podemos usar imágenes de fondo. En este caso, tenemos que declarar el fondo con la propiedad background-image y declarar la URL de la imagen con la función url(), como lo hacemos en el siguiente ejemplo.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  background-image: url("ladrillosclaros.jpg");
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-48: *Agregando una imagen de fondo*

El problema con las imágenes es que no siempre son del mismo tamaño que la caja creada por el elemento. Por esta razón, los navegadores repiten la imagen una y otra vez hasta cubrir toda el área. El resultado se muestra en la Figura 3-22.

Figura 3-22: *Imagen de fondo*

Ejercicio 3.8 :reemplace las reglas en su archivo CSS por las reglas del Listado 3-48. Descargue la imagen ladrillosclaros.jpg desde nuestro sitio web. Copie la imagen en el mismo directorio donde se encuentra su documento. Abra el documento en su navegador. Debería ver algo parecido a lo que se muestra en la Figura 3-22.

Si queremos modificar el comportamiento por defecto, podemos usar el resto de las propiedades de fondo disponibles. Por ejemplo, si asignamos el valor repeat-y a la propiedad background-repeat, el navegador solo repetirá la imagen en el eje vertical.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  background-image: url("ladrillosclaros.jpg");
  background-repeat: repeat-y;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-49: *Configurando el fondo*

Figura 3-23: *Fondo de imagen*

Cuando nuestro diseño requiere varios valores para configurar el fondo, podemos declararlos todos juntos con la propiedad background. Esta propiedad nos permite declarar diferentes aspectos del fondo en una sola línea de código.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  background: #CCCCCC url("ladrillosclaros.jpg") repeat-y;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-50: *Configurando el fondo con la propiedad* `background`

La regla del Listado 3-50 especifica los mismos valores que usamos anteriormente, pero ahora combina una imagen de fondo con un color. El resultado se muestra en la Figura 3-24.

Figura 3-24: *Imagen de fondo combinada con un color de fondo*

Bordes

Los elementos pueden incluir un borde en los límites de la caja del elemento. Por defecto, los navegadores no muestran ningún borde, pero podemos usar las siguientes propiedades para definirlo.

border-width—Esta propiedad define el ancho del borde. Acepta hasta cuatro valores separados por un espacio para especificar el ancho de cada lado del borde (superior, derecho, inferior, e izquierdo, es ese orden). También podemos declarar el ancho para cada lado de forma independiente con las propiedades **`border-top-width`**, **`border-bottom-width`**, **`border-left-width`**, y **`border-right-width`**.

border-style—Esta propiedad define el estilo del borde. Acepta hasta cuatro valores separados por un espacio para especificar los estilos de cada lado del borde (superior, derecho, inferior, e izquierdo, en ese orden). Los valores disponibles son **`none`**, **`hidden`**, **`dotted`**, **`dashed`**, **`solid`**, **`double`**, **`groove`**, **`ridge`**, **`inset`**, y **`outset`**. El valor por defecto es **`none`**, lo que significa que el borde no se mostrará a menos que asignemos un valor diferente a esta propiedad. También podemos declarar los estilos de forma independiente con las propiedades **`border-top-style`**, **`border-bottom-style`**, **`border-left-style`**, y **`border-right-style`**.

border-color—Esta propiedad define el color del borde. Acepta hasta cuatro valores separados por un espacio para especificar el color de cada lado del borde (superior, derecho, inferior, e izquierdo, en ese orden). También podemos declarar los colores de forma independiente con las propiedades **`border-top-color`**, **`border-bottom-color`**, **`border-left-color`**, y **`border-right-color`**.

border—Esta propiedad nos permite declarar todos los atributos del borde al mismo tiempo. También podemos usar las propiedades **`border-top`**, **`border-bottom`**, **`border-left`**, y **`border-right`** para definir los valores de cada borde de forma independiente.

Para asignar un borde a un elemento, todo lo que tenemos que hacer es definir el estilo con la propiedad border-style. Una vez que se define el estilo, el navegador usa los valores por defecto para generar el borde. Si no queremos dejar que el navegador determine estos valores, podemos usar el resto de las propiedades para configurar todos los atributos del borde. El siguiente ejemplo asigna un borde sólido de 2 píxeles de ancho a la cabecera de nuestro documento.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border-style: solid;
  border-width: 2px;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-51: *Agregando un borde a un elemento*

Figura 3-25: *Borde sólido*

Como hemos hecho con la propiedad background, declaramos todos los valores juntos usando la propiedad border. El siguiente ejemplo crea un borde discontinuo de 5 píxeles en color gris.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 5px dashed #CCCCCC;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-52: *Definiendo un borde con la propiedad* `border`

Figura 3-26: *Borde discontinuo*

El borde agregado con estas propiedades se dibuja alrededor de la caja del elemento, lo que significa que va a describir un rectángulo con esquinas rectas. Si nuestro diseño requiere esquinas redondeadas, podemos agregar la siguiente propiedad.

border-radius—Esta propiedad define el radio del círculo virtual que el navegador utilizará para dibujar las esquinas redondeadas. Acepta hasta cuatro valores para definir los radios de cada esquina (superior izquierdo, superior derecho, inferior derecho e inferior izquierdo, en ese orden). También podemos usar las propiedades **`border-top-left-radius`**, **`border-top-right-radius`**, **`border-bottom-right-radius`**, y **`border-bottom-left-radius`** para definir el radio de cada esquina de forma independiente.

El siguiente ejemplo genera esquinas redondeadas para nuestra cabecera con un radio de 20 píxeles.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 2px solid;
  border-radius: 20px;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-53: *Generando esquinas redondeadas*

Figura 3-27: *Esquinas redondeadas*

Si todas las esquinas son iguales, podemos declarar solo un valor para esta propiedad, tal como lo hemos hecho en el ejemplo anterior. Sin embargo, al igual que con las propiedades margin y padding, si queremos que las esquinas sean diferentes, tenemos que especificar valores diferentes para cada una de ellas.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 2px solid;
  border-radius: 20px 10px 30px 50px;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-54: *Declarando diferentes valores para cada esquina*

En el Listado 3-54, los valores asignados a la propiedad border-radius representan cuatro ubicaciones diferentes. Los valores se declaran siempre en la dirección de las agujas del reloj, comenzando por la esquina superior izquierda. El orden es: esquina superior izquierda, esquina superior derecha, esquina inferior derecha y esquina inferior izquierda.

Figura 3-28: *Diferentes esquinas*

Lo básico: al igual que las propiedades **`margin`** y **`padding`**, la propiedad **`border-radius`** también puede aceptar solo dos valores. El primer valor se asigna a las esquinas primera y tercera (superior izquierdo e inferior derecho) y a las esquinas segunda y cuarta (superior derecho e inferior izquierdo).

También podemos cambiar la forma de las esquinas agregando valores separados por una barra oblicua. Los valores de la izquierda representan el radio horizontal y los valores de la derecha representan el radio vertical. La combinación de estos valores genera una elipse.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 2px solid;
  border-radius: 20px / 10px;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-55: *Generando esquinas elípticas*

Figura 3-29: *Esquinas elípticas*

Ejercicio 3.9 :copie los estilos que desea probar dentro de su archivo CSS y abra el documento en su navegador. Modifique los valores de cada propiedad pare entender cómo trabajan.

Los bordes que acabamos de crear se dibujan en los límites de la caja del elemento, pero también podemos demarcar el elemento con un segundo borde que se dibuja alejado de esto límites. El propósito de estos tipos de bordes es resaltar el elemento. Algunos navegadores lo usan para demarcar texto, pero la mayoría dibuja un segundo borde fuera de los límites de la caja. CSS ofrece las siguientes propiedades para crear este segundo borde.

outline-width—Esta propiedad define el ancho del borde. Acepta valores en cualquiera de las unidades disponibles en CSS (**px**, **%**, **em**, etc.) y también las palabras clave **thin**, **medium**, y **thick**.

outline-style—Esta propiedad define el estilo del borde. Los valores disponibles son **none**, **auto**, **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset**, y **outset**.

outline-color—Esta propiedad define el color del borde.

outline-offset—Esta propiedad define el desplazamiento (a qué distancia de los límites de la caja se dibujará el segundo borde). Acepta valores en cualquiera de las unidades disponibles en CSS (**px**, **%**, **em**, etc.).

outline—Esta propiedad nos permite especificar el ancho, estilo y color del borde al mismo tiempo (el desplazamiento aún se debe definir con la propiedad **outline-offset**).

Por defecto, el desplazamiento se declara con el valor 0, por lo que el segundo borde se dibujará a continuación del borde de la caja. Si queremos separar los dos bordes, tenemos que definir la propiedad outline-offset.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  outline: 2px dashed #000000;
  outline-offset: 15px;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-56: *Agregando un segundo borde a la cabecera*

En el Listado 3-56, agregamos un segundo borde de 2 píxeles con un desplazamiento de 15 píxeles a los estilos originales asignados a la caja de la cabecera de nuestro documento. El resultado se muestra en la Figura 3-30.

Figura 3-30: *Segundo borde*

Los efectos logrados por las propiedades border y outline se limitan a simples líneas y unas pocas opciones de configuración, pero CSS nos permite definir bordes personalizados usando imágenes para superar estas limitaciones. Las siguientes son las propiedades que se incluyen con este propósito.

border-image-source—Esta propiedad determina la imagen que se usará para crear el borde. La URL del archivo se declara con la función **`url()`** (por ejemplo, **`url("ladrillos.jpg")`**).

border-image-width—Esta propiedad define el ancho del borde. Acepta valores en cualquiera de las unidades disponibles en CSS (**`px`**, **`%`**, **`em`**, etc.).

border-image-repeat—Esta propiedad define cómo se usa la imagen para generar el borde. Los valores disponibles son **`repeat`**, **`round`**, **`stretch`**, y **`space`**.

border-image-slice—Esta propiedad define cómo se va a cortar la imagen para representar las esquinas del borde. Debemos asignar cuatro valores para especificar los cuatro trozos de la imagen que se utilizarán (si solo se declara un valor, se usa para todos los lados). El valor se puede especificar como un entero o en porcentaje.

border-image-outset—Esta propiedad define el desplazamiento del borde (la distancia a la que se encuentra de la caja del elemento). Acepta valores en cualquiera de las unidades disponibles en CSS (**`px`**, **`%`**, **`em`**, etc.).

border-image—Esta propiedad nos permite especificar todos los atributos del borde al mismo tiempo.

Estas propiedades usan una imagen como patrón. De acuerdo a los valores facilitados, la imagen se corta como un pastel para obtener las piezas necesarias y luego estas piezas se acomodan alrededor del elemento para construir el borde.

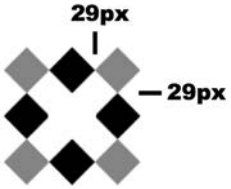

Figura 3-31: *Patrón para construir el borde*

Para lograr este objetivo, necesitamos declarar tres atributos: la ubicación del archivo de la imagen, el tamaño de las piezas que queremos extraer del patrón y las palabras clave que determinan cómo se van a distribuir estas piezas alrededor del elemento.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 29px solid;
  border-image-source: url("diamantes.png");
  border-image-slice: 29;
  border-image-repeat: stretch;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-57: *Creando un borde personalizado para la caja de la cabecera*

En el Listado 3-57, creamos un borde de 29 píxeles para la caja de la cabecera y luego cargamos la imagen diamantes.png para construir el borde. El valor 29 asignado a la propiedad border-image-slice declara el tamaño de las piezas, y el valor stretch, asignado a la propiedad border-image-repeat, es uno de los métodos disponibles para distribuir estas piezas alrededor de la caja. Hay tres valores disponibles para este último atributo. El valor repeat repite las piezas tomadas de la imagen cuantas veces sea necesario para cubrir el lado del elemento. En este caso, el tamaño de la pieza se preserva y la imagen solo se corta si no hay espacio suficiente para ubicarla. El valor round calcula la longitud del lado de la caja y luego estira las piezas para asegurarse de que no se corta ninguna. Finalmente, el valor stretch (usado en el Listado 3-57) estira una pieza hasta cubrir todo el lado.

Figura 3-32: *Borde de tipo* `stretch`

Como siempre, podemos declarar todos los valores al mismo tiempo usando una sola propiedad.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 29px solid;
  border-image: url("diamantes.png") 29 round;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-58: *Definiendo el borde con la propiedad* `border-image`

Figura 3-33: *Borde de tipo* `round`

Sombras

Otro efecto interesante que podemos aplicar a un elemento son las sombras. CSS incluye las siguientes propiedades para generar sombras para la caja de un elemento y también para formas irregulares como texto.

box-shadow—Esta propiedad genera una sombra desde la caja del elemento. Acepta hasta seis valores. Podemos declarar el desplazamiento horizontal y vertical de la sombra, el radio de difuminado, el valor de propagación, el color de la sombra, y también podemos incluir el valor **`inset`** para indicar que la sombra deberá proyectarse dentro de la caja.

text-shadow—Esta propiedad genera una sombra desde un texto. Acepta hasta cuatro valores. Podemos declarar el desplazamiento horizontal y vertical, el radio de difuminado y el color de la sombra.

La propiedad box-shadow necesita al menos tres valores para poder determinar el color y el desplazamiento de la sombra. El desplazamiento puede ser positivo o negativo. Los valores indican la distancia horizontal y vertical desde la sombra al elemento: los valores negativos posicionan la sombra a la izquierda y encima del elemento, mientras que los positivos crean una sombra a la derecha y debajo del elemento. El valor 0 ubica a la sombra detrás del elemento y ofrece la posibilidad de generar un efecto de difuminado a su alrededor. El siguiente ejemplo agrega una sombra básica a la cabecera de nuestro documento.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  box-shadow: rgb(150,150,150) 5px 5px;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-59: *Agregando una sombra a la cabecera*

Figura 3-34: *Sombra básica*

La sombra que hemos obtenido hasta el momento es sólida, sin gradientes ni transparencia, pero aún no se parece a una sombre real. Para mejorar su aspecto podemos agregar una distancia de difuminado.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  box-shadow: rgb(150,150,150) 5px 5px 20px;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-60: *Agregando el valor de difuminado con la propiedad* `box-shadow`

Figura 3-35: *Sombra*

Al agregar otro valor en píxeles al final de la propiedad, podemos propagar la sombra. Este efecto cambia levemente la apariencia de la sombra y expande el área que ocupa.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  box-shadow: rgb(150,150,150) 10px 10px 20px 10px;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-61: *Expandiendo la sombra*

Figura 3-36: *Sombra más amplia*

El último valor disponible para la propiedad box-shadow no es un número, sino la palabra clave inset. Este valor transforma la sombra externa en una sombra interna, lo cual otorga un efecto de profundidad al elemento.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  box-shadow: rgb(150,150,150) 5px 5px 10px inset;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-62: *Creando una sombra interna*

Figura 3-37: *Sombra interna*

IMPORTANTE: las sombras no expanden el elemento ni tampoco incrementan su tamaño, por lo que deberá asegurarse de que haya suficiente espacio alrededor del elemento para que se vea la sombra.

La propiedad box-shadow se ha diseñado específicamente para cajas de elementos. Si intentamos aplicar este efecto a un elemento , por ejemplo, la sombra se asignará a la caja alrededor del elemento, no a su contenido. CSS define una propiedad aparte para generar la sombra de un texto llamada text-shadow.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
}
#titulo {
  font: bold 26px Verdana, sans-serif;
  text-shadow: rgb(150, 150, 150) 3px 3px 5px;
}
```

Listado 3-63: *Agregando una sombra al título*

Los valores de la propiedad text-shadow son similares a los asignados a la propiedad box-shadow. Podemos especificar el color de la sombra, la distancia horizontal y vertical desde la sombra al elemento, y el radio de difuminado. En el Listado 3-63, se genera una sombra para el título de la cabecera con una distancia de 3 píxeles y un radio de difuminado de 5. El resultado se muestra en la Figura 3-38.

Figura 3-38: *Sombra de texto*

Gradientes

Un gradiente se forma mediante una serie de colores que varían continuamente con una transición suave de un color a otro. Los gradientes se crean como imágenes y se agregan al fondo del elemento con las propiedades background-image o background. Para crear la imagen con el gradiente, CSS ofrece las siguientes funciones:

linear-gradient(posición, ángulo, colores**)**—Esta función crea un gradiente lineal. El atributo **posición** determina el lado o la esquina desde la cual comienza el gradiente y se declara con los valores **`top`**, **`bottom`**, **`left`** y **`right`**; el atributo **ángulo** define la dirección del gradiente y se puede declarar en las unidades **`deg`** (grados), **`grad`** (gradianes), **`rad`** (radianes), o **`turn`** (espiras), y el atributo **colores** es la lista de colores que participan en el gradiente separados por coma. Los valores para el atributo **colores** pueden incluir un segundo valor en porcentaje separado por un espacio para indicar la posición donde finaliza el color.

radial-gradient(posición, forma, colores, extensión**)**—Esta función crea un gradiente radial. El atributo **posición** indica el origen del gradiente y se puede declarar en píxeles, en porcentaje, o por medio de la combinación de los valores **`center`**, **`top`**, **`bottom`**, **`left`** y **`right`**, el atributo **forma** determina la forma del gradiente y se declara con los valores **`circle`** y **`ellipse`**, el atributo **colores** es la lista de los colores que participan en el gradiente separados por coma, y el atributo **extensión** determina la forma que el gradiente va a adquirir con los valores **`closest-side`**, **`closest-corner`**, **`farthest-side`**, y **`farthest-corner`**. Los valores para el atributo **colores** pueden incluir un segundo valor en porcentaje separado por un espacio para indicar la posición donde finaliza el color.

Los gradientes se declaran como imágenes de fondo, por lo que podemos aplicarlos a un elemento por medio de las propiedades background-image o background, como lo hacemos en el siguiente ejemplo.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  background: -webkit-linear-gradient(top, #FFFFFF, #666666);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-64: *Agregando un gradiente lineal a la cabecera*

Figura 3-39: *Gradiente lineal*

IMPORTANTE: algunas propiedades y funciones CSS todavía se consideran experimentales. Por esta razón, se deben declarar con un prefijo que representa el motor web utilizado. Por ejemplo, si queremos que la función **`linear-gradient()`** funcione en Google Chrome, tenemos que declararla como **`-webkit-linear-gradient()`**. Si quiere usar esta función en su sitio web, deberá repetir la propiedad **`background`** para cada navegador existente con su correspondiente prefijo. Los prefijos requeridos para los navegadores más populares son **`-moz-`** para Mozilla Firefox, **`-webkit-`** para Safari y Google Chrome, **`-o-`** para Opera, y **`-ms-`** para Internet Explorer.

En el ejemplo anterior, usamos el valor top para determinar la posición inicial del gradiente, pero también podemos combinar dos valores para comenzar el gradiente desde una esquina del elemento, como en el siguiente ejemplo.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  background: -webkit-linear-gradient(top right, #FFFFFF, #666666);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-65: *Estableciendo la posición inicial*

Figura 3-40: *Diferente comienzo para un gradiente lineal*

Cuando trabajamos con gradientes lineales, también podemos configurar la dirección con un ángulo en grados.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  background: -webkit-linear-gradient(30deg, #FFFFFF, #666666);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-66: *Creando un gradiente con una dirección de 30 grados*

Figura 3-41: *Gradiente lineal con la dirección establecida en grados*

Los gradientes anteriores se han creado con solo dos colores, pero podemos agregar más valores para generar un gradiente multicolor.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  background: -webkit-linear-gradient(top, #000000, #FFFFFF, #999999);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-67: *Creando un gradiente multicolor*

Figura 3-42: *Gradiente lineal multicolor*

Si usamos el valor transparent en lugar de un color, podemos hacer que el gradiente sea traslúcido y de esta manera se mezcle con el fondo (este efecto también se puede lograr con la función rgba() estudiada anteriormente). En el siguiente ejemplo, asignamos una imagen de fondo al elemento <body> para cambiar el fondo de la página y poder comprobar que logramos un gradiente translúcido.

```
body {
  background: url("ladrillosclaros.jpg");
}
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  background: -webkit-linear-gradient(top, transparent, #666666);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-68: *Creando un gradiente translúcido*

Figura 3-43: *Gradiente translúcido*

Los parámetros que definen los colores también pueden determinar el punto de comienzo y final de cada color incluyendo un valor adicional en porcentaje.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  background: -webkit-linear-gradient(top, #FFFFFF 50%, #666666 90%);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-69: *Configurando los puntos de comienzo y final de cada color*

Figura 3-44: *Gradiente lineal con valores de comienzo y final*

Además de gradientes lineales, también podemos crear gradientes radiales. La sintaxis para gradientes radiales no difiere mucho de los gradientes lineales que acabamos de estudiar. La única diferencia es que tenemos que usar la función radial-gradient() en lugar de la función linear-gradient() e incluir un parámetro que determina la forma del gradiente con los valores circle o ellipse.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  background: -webkit-radial-gradient(center, ellipse, #FFFFFF,
#000000);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-70: *Creando un gradiente radial*

Figura 3-45: *Gradiente radial*

Excepto por la forma (círculo o elipse), el resto de esta función trabaja del mismo modo que linear-gradient(). La posición del gradiente se puede personalizar y podemos usar varios colores con un segundo valor para determinar los límites de cada uno de ellos.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  background: -webkit-radial-gradient(30px 50px, ellipse, #FFFFFF 50%,
#666666 70%, #999999 90%);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-71: *Creando un gradiente radial multicolor*

Figura 3-46: *Gradiente radial con puntos de inicio y final*

Filtros

Los filtros agregan efectos a un elemento y su contenido. CSS incluye la propiedad filter para asignar un filtro a un elemento y las siguiente funciones para crearlo.

blur(valor**)**—Esta función produce un **efecto de difuminado**. Acepta valores en píxeles desde `1px` a `10px`.

grayscale(value**)**—Esta función convierte los colores de la imagen en una **escala de grises**. Acepta números decimales entre `0.1` y `1`.

drop-shadow(x, y, tamaño, color**)**—Esta función genera una **sombra**. Los atributos **x** e **y** determinan la distancia entre la sombra y la imagen, el atributo **tamaño** especifica el tamaño de la sombra, y el atributo **color** declara su color.

sepia(valor**)**—Esta función le otorga un **tono sepia** (ocre) a los colores de la imagen. Acepta números decimales entre `0.1` y `1`.

brightness(valor**)**—Esta función cambia el **brillo de la imagen**. Acepta números decimales entre `0.1` y `10`.

contrast(valor**)**—Esta función cambia el **contraste de la imagen**. Acepta números decimales entre `0.1` y `10`.

hue-rotate(valor**)**—Esta función aplica una rotación a los matices de la imagen. Acepta un valor en grados desde `1deg` a `360deg`.

invert(valor**)**—Esta función invierte los colores de la imagen y produce un **negativo**. Acepta números decimales entre `0.1` y `1`.

saturate(valor**)**—Esta función **satura los colores** de la imagen. Acepta números decimales entre `0.1` y `10`.

opacity(valor**)**—Esta función cambia la **opacidad** de la imagen. Acepta números decimales entre `0` y `1` (`0` es totalmente transparente y `1` totalmente opaco).

Estos filtros no solo se pueden aplicar a imágenes, sino también a otros elementos en el documento. El siguiente ejemplo aplica un efecto de difuminado a la cabecera de nuestro documento.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  filter: blur(5px);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-72: *Aplicando un filtro a la cabecera*

Figura 3-47: *Una cabecera borrosa*

Ejercicio 3.10 :reemplace las reglas en su archivo CSS por las reglas delListado 3-72 y abra el documento en su navegador. Reemplace la función **`blur()`** con cualquiera de las restantes funciones disponibles para ver cómo trabajan.

Transformaciones

Una vez que se crean los elementos HTML, estos permanecen inmóviles, pero podemos modificar su posición, tamaño y apariencia por medio de la propiedad transform. Esta propiedad realiza cuatro transformaciones básicas a un elemento: escalado, rotación, inclinación y traslación. Las siguientes son las funciones definidas para este propósito.

scale(x, y**)**—Esta función **modifica la escala del elemento**. Existen otras dos funciones relacionadas llamadas **`scaleX()`** y **`scaleY()`** pare especificar los valores horizontales y verticales independientemente.

rotate(ángulo**)**—Esta función **rota el elemento**. El atributo representa los grados de rotación y se puede declarar en **`deg`** (grados), **`grad`** (gradianes), **`rad`** (radianes) o **`turn`** (espiras).

skew(ángulo**)**—Esta función **inclina el elemento**. El atributo representa los grados de desplazamiento. La función puede incluir dos valores para representar el ángulo horizontal y vertical. Los valores se pueden declarar en **`deg`** (grados), **`grad`** (gradianes), **`rad`** (radianes) o **`turn`** (espiras).

translate(x, y**)**—Esta función **desplaza al elemento** a la posición determinada por los atributos **x** e **y**.

La función scale() recibe dos parámetros, el valor x para la escala horizontal y el valor y para la escala vertical. Si solo se declara un valor, ese mismo valor se usa para ambos parámetros.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  transform: scale(2);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-73: *Escalando la cabecera*

En el ejemplo del Listado 3-73, transformamos la cabecera con una escala que duplica su tamaño. A la escala se le pueden asignar números enteros y decimales, y esta escala se calcula por medio de una matriz. Los valores entre 0 y 1 reducen el tamaño del elemento, el valor 1 preserva las proporciones originales, mientras que los valores sobre 1 incrementan las dimensiones del elemento de forma lineal.

Cuando asignamos valores negativos a esta función, se genera un efecto interesante.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  transform: scale(1, -1);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-74: *Creando una imagen espejo con la función* `scale()`

En el Listado 3-74, se declaran dos parámetros para modificar la escala de la cabecera. Ambos valores preservan las proporciones originales, pero el segundo valor es negativo y, por lo tanto, invierte el elemento en el eje vertical, produciendo una imagen invertida.

Figura 3-48: *Imagen espejo con* `scale()`

Además de escalar el elemento, también podemos rotarlo con la función rotate(). En este caso, los valores negativos cambian la dirección en la cual se rota el elemento. El siguiente ejemplo rota la cabecera 30 grados en el sentido de las agujas del reloj.

```
header {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  transform: rotate(30deg);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-75: *Rotando la cabecera*

Otra función disponible para la propiedad transform es skew(). Esta función cambia la simetría del elemento en grados y en una o ambas dimensiones.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  transform: skew(20deg);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-76: *Inclinando la cabecera*

Esta función recibe dos valores, pero a diferencia de otras funciones, cada parámetro solo afecta a una dimensión (los parámetros son independientes). En el Listado 3-76, solo el primer parámetro se declara y, por lo tanto, solo se modifica la dimensión horizontal. Si lo deseamos, podemos usar las funciones adicionales skewX() y skewY() para lograr el mismo efecto.

Figura 3-49: *Inclinación horizontal*

La pantalla de un dispositivo se divide en filas y columnas de píxeles (la mínima unidad visual de la pantalla). Con el propósito de identificar la posición de cada píxel, los ordenadores usan un sistema de coordenadas, donde las filas y columnas de píxeles se cuentan desde la esquina superior izquierda a la esquina inferior derecha de la cuadrícula, comenzando por el valor 0 (los valores se incrementan de izquierda a derecha y de arriba abajo). Por ejemplo, el primer píxel de la esquina superior izquierda de la pantalla se encuentra en la posición 0,0 (columna 0, fila 0), mientras que un píxel que se encuentra 30 píxeles del lado izquierdo de la pantalla y 10 píxeles de la parte superior estará ubicado en la posición 30,10.

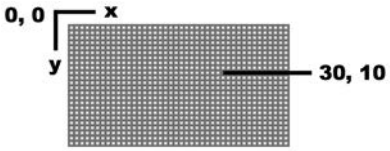

Figura 3-50: *Sistema de coordenadas*

El punto en la posición 0,0 se llama origen, y las líneas de columnas y filas se denominan ejes y se identifican con las letras x e y (x para columnas e y para filas), tal como ilustra la Figura 3-50. Usando la propiedad transform podemos cambiar la posición de un elemento en esta cuadrícula. La función que tenemos que asignar a la propiedad en este caso es translate().

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  transform: translate(100px);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-77: *Moviendo la cabecera hacia la derecha*

La función translate() utiliza el sistema de coordenadas para establecer la posición del elemento, usando su posición actual como referencia. La esquina superior izquierda del elemento se considera que está en la posición 0,0, por lo que los valores negativos mueven el elemento hacia la izquierda o encima de la posición original, y los valores positivos lo mueven a la derecha y abajo.

En el Listado 3-77, movemos la cabecera a la derecha 100 píxeles de su posición original. Se pueden declarar dos valores en esta función para mover el elemento horizontalmente y verticalmente, o podemos usar las funciones translateX() y translateY() para declarar los valores de forma independiente.

A veces, puede resultar útil aplicar varias transformaciones a la vez a un mismo elemento. Para combinar transformaciones con la propiedad transform, tenemos que declarar las funciones separadas por un espacio.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  transform: translateY(100px) rotate(45deg) scaleX(0.3);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-78: *Moviendo, escalando y rotando el elemento con una sola propiedad*

IMPORTANTE: cuando combinamos múltiples funciones, debemos considerar que el orden de combinación es importante. Esto se debe a que algunas funciones modifican el punto de origen y el centro del elemento y, por lo tanto, cambian los parámetros sobre los que el resto de las funciones trabajan.

De la misma manera que podemos generar transformaciones en dos dimensiones sobre elementos HTML, también podemos hacerlo en tres dimensiones. Estos tipos de transformaciones se realizan considerando un tercer eje que representa profundidad y se identifica con la letra z. Las siguientes son las funciones disponibles para este propósito.

scale3d(x, y, z**)**—Esta función **asigna una nueva escala al elemento en un espacio 3D**. Acepta tres valores en números decimales para establecer la escala en los ejes **x**, **y** y **z**. Al igual que con transformaciones 2D, el valor 1 preserva la escala original.

rotate3d(x, y, z, ángulo**)**—Esta función **rota el elemento en un ángulo** y sobre un eje específicos. Los valores para los ejes se deben especificar en números decimales y el ángulo se puede expresar en **`deg`** (grados), **`grad`** (gradianes), **`rad`** (radianes), o **`turn`** (espiras). Los valores asignados a los ejes determinan un vector de rotación, por lo que los valores no son importantes, pero sí lo es la relación entre los mismos. Por ejemplo, **`rotate3d(5, 2, 6, 30deg)`** producirá el mismo efecto que **`rotate3d(50, 20, 60, 30deg)`**, debido a que el vector resultante es el mismo.

translate3d(x, y, z**)**—Esta función **mueve el elemento a una nueva posición en el espacio 3D**. Acepta tres valores en píxeles para los ejes **x**, **y** y **z**.

perspective(valor**)**—Esta función **agrega un efecto de profundidad** a la escena incrementando el tamaño del lado del elemento cercano al espectador.

Algunas transformaciones 3D se pueden aplicar directamente al elemento, como hemos hecho con las transformaciones 2D, pero otras requieren que primero declaremos la perspectiva. Por ejemplo, si rotamos el elemento en el eje y, un lado del elemento se desplazará hacia adelante y el otro hacia atrás, pero el tamaño de cada lado permanecerá igual y por ello el usuario no verá la transformación. Para lograr un efecto realista, tenemos que declarar la perspectiva con la función perspective().

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;

  border: 1px solid;
  transform: perspective(500px) rotate3d(0, 1, 0, 45deg);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-79: *Aplicando un efecto tridimensional a la cabecera*

La regla del Listado 3-79 primero asigna el valor de la perspectiva y luego rota el elemento 45 grados en el eje y (para seleccionar un eje, tenemos que declarar al resto de los ejes con el valor 0). El navegador rota el elemento y, debido a que definimos la perspectiva para la transformación, expande un lado del elemento y reduce el otro para crear la impresión de perspectiva.

Figura 3-51: *Efecto 3D con perspectiva*

CSS también incluye algunas propiedades que podemos usar para lograr un efecto más realista.

perspective—Esta propiedad trabaja de forma similar a la función **`perspective()`**, pero opera en el elemento padre. La propiedad crea un contenedor que aplica el efecto de perspectiva a los elementos en su interior.

perspective-origin—Esta propiedad cambia las coordenadas **`x`** e **`y`** del espectador. Acepta dos valores en porcentaje, píxeles, o las palabras clave **`center`**, **`left`**, **`right`**, **`top`** y **`bottom`**. Los valores por defecto son **`50% 50%`**.

backface-visibility—Esta propiedad determina si el reverso del elemento será visible o no. Acepta dos valores: **`visible`** o **`hidden`**, con el valor **`visible`** configurado por defecto.

Debido a que en nuestro ejemplo la cabecera del documento es hija directa del cuerpo, tenemos que asignar estas propiedades al elemento <body>. El siguiente ejemplo define la perspectiva del cuerpo y luego rota la cabecera con la propiedad transform, como hemos hecho anteriormente.

```
body {
  perspective: 500px;
  perspective-origin: 50% 90%;
}
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  transform: rotate3d(0, 1, 0, 135deg);
}

#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-80: *Declarando un origen diferente para el espectador*

El resultado de aplicar la perspectiva al elemento padre es el mismo que si usáramos la función perspective() directamente en el elemento que queremos modificar, pero declarando la perspectiva de esta manera podemos usar la propiedad perspective-origin y al mismo tiempo cambiar las coordenadas del espectador.

Figura 3-52: *Efecto 3D usando la propiedad* `perspective`

Ejercicio 3.11 :reemplace las reglas en su archivo CSS por las reglas del Listado 3-80 y abra el documento en su navegador. Agregue la propiedad **`backface-visibility`** con el valor **`hidden`** a la cabecera para volverla invisible cuando está invertida.

Todas las funciones que hemos estudiado hasta el momento modifican los elementos en el documento, pero una vez que la página web se muestra, permanece igual. Sin embargo, podemos combinar transformaciones y seudoclases para convertir nuestro documento en una aplicación dinámica. La seudoclase que podemos implementar en este caso se llama :hover. Las seudoclases, como hemos visto anteriormente, agregan efectos especiales a las reglas. En este caso, la regla con la seudoclase :hover se aplica solo cuando el ratón se encuentra sobre el elemento que referencia.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
}
header:hover {
  transform: rotate(5deg);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-81: *Respondiendo al ratón*

En el Listado 3-81, la regla original de la cabecera es la misma, pero ahora se agrega una nueva regla identificada con el selector header:hover para aplicar un efecto de transformación cuando el ratón se encuentra sobre el elemento. En consecuencia, cada vez que el usuario mueve el ratón sobre la cabecera, la propiedad transform rota el elemento 5 grados, y cuando el puntero se mueve fuera de la caja del elemento, el mismo se rota nuevamente a su posición original. Este código logra una animación básica pero útil usando solo propiedades CSS.

Transiciones

Con la seudoclase :hover podemos realizar transformaciones dinámicas. Sin embargo, una animación real requiere una transición entre los dos pasos del proceso. Para este propósito, CSS ofrece las siguientes propiedades.

transition-property—Esta propiedad especifica las propiedades que participan en la transición. Además de los nombres de las propiedades, podemos asignar el valor **`all`** para indicar que todas las propiedades participarán de la transición.

transition-duration—Esta propiedad especifica la **duración de la transición** en segundos (**`s`**).

transition-timing-function—Esta propiedad determina la función que se usa para calcular los valores para la transición. Los valores disponibles son **`ease`**, **`ease-in`**, **`ease-out`**, **`ease-in-out`**, **`linear`**, **`step-start`**, y **`step-end`**.

transition-delay—Esta propiedad especifica el tiempo que el n**avegador esperará antes de iniciar la animación**.

transition—Esta propiedad nos permite declarar **todos los valores de la transición al mismo tiempo.**

Implementando estas propiedades le indicamos al navegador que tiene que crear todos los pasos de la animación y generar una transición entre el estado actual del elemento y el especificado por las propiedades. Las siguientes reglas implementan la propiedad transition para animar la transformación introducida en el ejemplo anterior.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  transition: transform 1s ease-in-out 0s;
}
header:hover {
  transform: rotate(5deg);
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-82: *Creando una animación con la propiedad* `transition`

La propiedad transition puede recibir hasta cuatro parámetros separados por un espacio. El primer valor es la propiedad que se considerará para crear la transición (en nuestro ejemplo, usamos la propiedad transform), el segundo parámetro determina la duración de la animación (1 segundo), el tercer parámetro es un valor que determina cómo se llevará a cabo la transición por medio de una curva Bézier, y el último parámetro determina cuántos segundos tarda la animación en comenzar.

Si la transición tiene que considerar los valores de más de una propiedad, tenemos que declarar los nombres de las propiedades separadas por coma. Cuando se tienen que considerar todas las propiedades que se modifican para crear la animación, podemos usar el valor all en su lugar.

Ejercicio 3.12 :reemplace las reglas en su archivo CSS por las reglas del Listado 3-82 y abra el documento en su navegador. Mueva el puntero del ratón sobre la cabecera para iniciar la animación.

Animaciones

La propiedad transition crea una animación básica, pero solo se involucran dos estados en el proceso: el estado inicial determinado por los valores actuales de las propiedades y el estado final, determinado por los nuevos valores. Para crear una animación real, necesitamos declarar más de dos estados, como los fotogramas de una película. CSS incluye las siguientes propiedades para componer animaciones más complejas.

animation-name—Esta propiedad **especifica el nombre** usado para identificar los pasos de la animación. Se puede usar para configurar varias animaciones al mismo tiempo declarando los nombres separados por coma.

animation-duration—Esta propiedad determina la **duración** de cada ciclo de la animación. El valor se debe especificar en segundos (por ejemplo, `1s`).

animation-timing-function—Esta propiedad determina **cómo se llevará a cabo** el proceso de animación por medio de una curva Bézier declarada con los valores `ease`, `linear`, `ease-in`, `ease-out` y `ease-in-out`.

animation-delay—Esta propiedad especifica el **tiempo que el navegador esperará antesde iniciar la animación.**

animation-iteration-count—Esta propiedad declara la **cantidad de veces** que se ejecutará la animación. Acepta un número entero o el valor `infinite`, el cual hace que la animación se ejecute por tiempo indefinido. El valor por defecto es `1`.

animation-direction—Esta propiedad declara la **dirección de la animación**. Acepta cuatro valores: `normal` (por defecto), `reverse`, `alternate`, y `alternate-reverse`. El valor `reverse` invierte la dirección de la animación, mostrando los pasos en la dirección opuesta en la que se han declarado. El valor `alternate` mezcla los ciclos de la animación, reproduciendo los que tienen un índice impar en dirección normal y el resto en dirección invertida. Finalmente, el valor `alternate-reverse` hace lo mismo que `alternate`, pero en sentido inverso.

animation-fill-mode—Esta propiedad define **cómo afecta** la animación a los estilos del elemento. Acepta los valores `none` (por defecto), `forwards`, `backwards`, y `both`. El valor `forwards` mantiene al elemento con los estilos definidos por las propiedades aplicadas en el último paso de la animación, mientras que `backwards` aplica los estilos del primer paso tan pronto como se define la animación (antes de ser ejecutada). Finalmente, el valor `both` produce ambos efectos.

animation—Esta propiedad nos **permite definir todos los valores** de la animación al mismo tiempo.

Estas propiedades configuran la animación, pero los pasos se declaran por medio de la regla @keyframes. Esta regla se debe identificar con el nombre usado para configurar la animación, y debe incluir la lista de propiedades que queremos modificar en cada paso. La posición de cada paso de la animación se determina con un valor en porcentaje, donde 0 % corresponde al primer fotograma o al comienzo de la animación, y 100 % corresponde al final.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  animation: mianimacion 1s ease-in-out 0s infinite normal none;
}
@keyframes mianimacion {
  0% {
    background: #FFFFFF;
  }
  50% {
    background: #FF0000;
  }
  100% {
    background: #FFFFFF;
  }
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-83: *Creando una animación compleja*

Las reglas del Listado 3-83 crean una animación que cambia los colores del fondo de la cabecera de rojo a blanco. La animación se ha definido mediante la propiedad animation con una duración de 1 segundo, y configurado para ejecutarse una y otra vez con el valor infinite. La propiedad también asigna el nombre mianimacion a la animación para poder configurar luego los pasos con la regla @keyframes.

Las propiedades indican en cada paso cómo será afectará al elemento. En este caso, declaramos tres pasos, uno al 0 %, otro al 50 %, y un tercero al 100 %. Cuando se inicia la animación, el navegador asigna al elemento los estilos definidos al 0 % y luego cambia los valores de las propiedades gradualmente hasta llegar a los valores definidos al 50 %. El proceso se repite desde este valor hasta el valor final asignado a la propiedad en el último paso (100 %).

En este ejemplo, definimos el estado inicial de la animación al 0 % y el estado final al 100 %, pero también podemos iniciar la animación en cualquier otro punto y declarar todos los pasos que necesitemos, como en el siguiente ejemplo.

```
header  {
  margin: 30px;
  padding: 15px;
  text-align: center;
  border: 1px solid;
  animation: mianimacion 1s ease-in-out 0s infinite normal none;
}
@keyframes mianimacion {
  20% {
    background: #FFFFFF;
  }
  35% {
    transform: scale(0.5);
    background: #FFFF00;
  }
  50% {
    transform: scale(1.5);
    background: #FF0000;
  }
  65% {
    transform: scale(0.5);
    background: #FFFF00;
  }
  80% {
    background: #FFFFFF;
  }
}
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

Listado 3-84: *Declarando más pasos para nuestra animación*

En el Listado 3-84, la animación comienza al 20 % y termina al 80 %, e incluye un total de cinco pasos. Cada paso de la animación modifica dos propiedades que incrementan el tamaño del elemento y cambian el color de fondo, excepto el primer paso y el último que solo cambian el color para lograr un efecto de rebote.