Introducción

CSS *(Cascading Style Sheets)* es un potente mecanismo para agregar estilo (colores, tipos de letra, espaciados, etc.) a las páginas web, pero no es un lenguaje de marcas. La pregunta que surge a continuación es obvia: ¿por qué tratar CSS en este libro?

Con el paso del tiempo y tras varias versiones, HTML ha dejado de contener información referente a cómo se deben presentar los datos en las páginas web, delegándose esta tarea en las hojas de estilo CSS. Solo con HTML las presentaciones de las páginas serían extremadamente austeras y carentes de atractivo. CSS está presente en casi todos los documentos HTML existentes, por lo que no se concibe una tecnología sin la otra. Por esta razón es necesario incluir CSS en este libro, ya que se puede considerar que forma con HTML un equipo inseparable.

3.1. Introducción, evolución y estado actual de CSS

CSS son las siglas de *Cascading Style Sheets* («hojas de estilo en cascada»), y es el lenguaje que se utiliza para definir el aspecto de las páginas HTML y XHTML.

Se diseñó para separar los datos (contenidos en los documentos HTML) de las reglas de presentación (contenidos en los documentos CSS), ya que, con anterioridad a su aparición, las reglas de presentación formaban parte de las etiquetas HTML.

Esta separación proporciona una enorme versatilidad a la hora de presentar la información, pues un mismo documento HTML puede combinarse con diferentes hojas de estilo CSS para generar distintas presentaciones.

En las figuras 3.1 y 3.2 se muestra el mismo documento HTML utilizando distintas hojas de estilo CSS. Las diferencias son evidentes: la tipografía, el color del texto, el color del fondo, la presencia o ausencia de las viñetas de la lista y la alineación del texto. Estos cambios se han introducido utilizando distintas hojas CSS sobre un mismo contenido HTML.

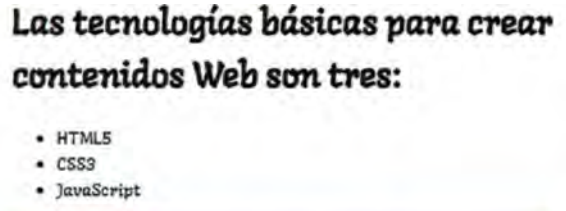

Figura 3.1. *Con CSS se asigna estilo a los documentos HTML.*

Figura 3.2. *Un mismo documento HTML puede tener múltiples presentaciones.*

Al igual que HTML, es el consorcio W3C la entidad encargada del mantenimiento y la estandarización de CSS. La primera versión CSS vio la luz en 1996 y desde entonces ha habido sucesivas modificaciones hasta llegar a la versión actual, denominada CSS3. No obstante, hay que aclarar algunos aspectos relacionados con las versiones. Hasta la versión 2.1, todas las anteriores se presentaban como una especificación única. Por cuestio-

nes de organización, el W3C decidió que la especificación de la versión 3 se realizaría por módulos, por lo que ya no existe una única versión actual, sino módulos actualizados a la versión 3 y módulos en versión 2.1. Sea como sea, en la jerga técnica se hace referencia a CSS3 como la versión actual de CSS, sin entrar en más detalles.

En la Tabla 3.1 se muestran las diferentes versiones y su año de publicación.

Tabla 3.1. Evolución de las versiones de CSS

AÑO	VERSIÓN	OBSERVACIONES
1996	CSS1	El W3C ya no mantiene esta versión.
1998	CSS2	El W3C ya no mantiene esta versión.
2011	CSS2.1	Corrige errores de CSS2.
2012	CSS3	Esta versión está dividida en módulos, por lo que no existe una única fecha global de cambio de versión.

Nota técnica

Todas las referencias realizadas en esta unidad a HTML afectan de manera implícita a XHTML, ya que el modelo de trabajo en relación con CSS es idéntico en ambos lenguajes.

3.2. Estructura y sintaxis de CSS

El funcionamiento de CSS consiste en definir unas reglas de presentación que se van a aplicar a un número indeterminado de elementos del documento HTML, al que están vinculadas. Por lo tanto, se necesitan dos herramientas básicas para crear y aplicar un estilo:

- **Selectores:** son las herramientas que permiten seleccionar el elemento o elementos sobre los que aplicar las reglas.
- **Declaraciones:** son las indicaciones para asignar, mediante pares de propiedad-valor, el aspecto deseado a los elementos determinados por el selector.

Los selectores y las declaraciones se agrupan en **reglas.**

En el siguiente ejemplo se muestra una regla compuesta por los distintos elementos:

- Selector **body.** El contenido se aplica a todos los elementos que se encuentren dentro del elemento **<body>** en la página HTML.
- Las declaraciones que asignan valor a las propiedades **text-align, font-family, background-color** y **color,** que afectarán respectivamente a la alineación del texto, el tipo de letra, el color de fondo y el color del elemento **<body>** y de todos los elementos contenidos en este, siempre que las propiedades sean aplicables a dichos elementos.

```
body {
        text-align: center;
        font-family: 'Roboto', sans-serif;
        background-color: black;
        color: white;
}
```

Los comentarios se delimitan con los símbolos /* y */ y no se procesan por el navegador.

```
/* Este texto es un comentario */
```

3.3. Aplicación de CSS

Para que las reglas CSS tengan efecto sobre los elementos de un documento HTML deben vincularse con este. Esto se puede realizar de tres maneras distintas:

- **Como un documento CSS externo.** Las reglas se almacenan en un documento externo con extensión .css y se establece una relación entre el documento HTML y dicho fichero. La asignación se hace mediante el elemento **<link>**, asignando «stysheet» como valor al atributo **rel** y la URI del fichero .css al atributo **href.**

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <link rel="stylesheet" href="ejemplo1_dark.css">
   <title>Título del documento</title>
</head>
<body>
   <H1>Las tecnologías básicas para crear contenidos Web son tres:</
H1>
   <ul>
      <li>HTML5</li>
      <li>CSS3</li>
      <li>JavaScript</li>
   </ul>
</body>
</html>
```

ejemplo1_dark.css

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

```
        background-color: black;
        color: white;
    }
    ul {
        list-style: none;
    }
```

- **Como una declaración CSS interna.** Las reglas se declaran en un elemento **<style>** incrustado en el propio documento HTML.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <style>
    body {
        text-align: center;
        background-color: black;
        color: white;
    }
    ul {
        list-style: none;
    }
    </style>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>
    <H1>Las tecnologías básicas para crear contenidos Web son tres:</
H1>
    <ul>
        <li>HTML5</li>
        <li>CSS3</li>
        <li>JavaScript</li>
    </ul>
</body>
</html>
```

- **Como una asignación *inline*** (en el propio elemento) de las reglas. Las reglas se declaran en el valor de la propiedad **style** del elemento al que afectan.

Ejemplo:

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

```
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Document</title>
</head>
<body style="text-align: center;background-color: black;color:white">
   <H1>Las tecnologías básicas para crear contenidos Web son tres:</
H1>
   <ul style="list-style: none;">
      <li>HTML5</li>
      <li>CSS3</li>
      <li>JavaScript</li>
   </ul>
</body>
</html>
```

Las tres alternativas no son exclusivas; se puede añadir una referencia a una o más hojas de estilo externas, incluir un elemento **<style>** y añadir un atributo **style** a un elemento. En ningún caso estas alternativas son incompatibles.

Cada una de las opciones para agregar estilos tiene sus ventajas e inconvenientes, siendo la más versátil y reutilizable la solución de crear un documento externo, ya que se podrá utilizar desde diferentes documentos. No obstante, las otras opciones a veces son elegidas por comodidad o para realizar prototipos rápidos, teniendo siempre en cuenta que las reglas no podrán reutilizarse con comodidad.

Actividad propuesta 3.1

Aplicación de estilo

Escribe en un editor los siguientes documentos HTML y CSS. Vincula el documento CSS con el documento HTML. A continuación abre el documento HTML con un navegador y observa el resultado.

Código HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Aplicando estilos</title>
    <!--AGREGAR EL VÍNCULO CON LA HOJA DE ESTILOS CSS-->
</head>
<body>
    <h1>Leonardo Da Vinci</h1>
    <p>El gran genio del <span class="ciudad">Renacimiento</span> italiano nació
en Anchiamo en el año 1452</p>
</body>
</html>
```

Código CSS:

```
h1 {
    text-transform: uppercase;
}
p {
    font-size: 1.5em;
    color: #888888;
}
.ciudad {
    font-style: italic;
    font-weight: bolder;
}
```

3.4. Prioridades y orden en CSS

El número de reglas que se puede aplicar a un documento HTML es ilimitado y estas pueden entrar en conflicto. Pendientes de enumerar y explicar los diferentes tipos de selectores, se puede adelantar que dos reglas, una que utiliza el selector **body** y otra que utiliza el selector **h1**, afectarán a los elementos **<h1>**. La pregunta que surge es ¿qué regla va a elegir el navegador en caso de que dos o más apliquen al mismo elemento? La respuesta está en la «C» de CSS.

Antes de explicar cómo se asignan las prioridades, hay que saber que pueden existir tres hojas de estilo distintas en una página: la propia del navegador, que es la primera que se aplica por defecto; la que proporciona el usuario, que se asocia al navegador, se aplica por defecto a todas las páginas y es muy útil para personas con discapacidad, y por último, están las hojas de estilo del diseñador, que se aplican al final y pueden ser más de una.

Hay tres conceptos fundamentales en la aplicación de los estilos: **cascada**, **especificidad** y **herencia**.

La **herencia** provoca que los valores de algunas de las propiedades aplicadas a un elemento contenedor se apliquen también a los elementos contenidos por él. No todas las propiedades se ven afectadas por este comportamiento, ya que en algunos casos no tendría sentido.

Los estilos CSS se aplican en **cascada**, según unas reglas fijas, obteniendo unos resultados predecibles. Los factores que participan en la decisión de qué estilo aplicar son los siguientes:

- **Origen e importancia:** los estilos inherentes al navegador tienen menos prioridad que los elegidos por el usuario y estos, a su vez, menos que los creados por el diseñador de la página, que son los que se acaban imponiendo en caso de conflicto. Dentro de los estilos proporcionados por el diseñador, influye si están definidos *inline* (mayor prioridad), en un elemento **<style>** (prioridad intermedia) o en un recurso externo (menor prioridad).
- **Nivel de especificidad:** cuanto más específico, mayor prioridad.

- **Orden de aparición:** los últimos estilos en leerse o procesarse se imponen a los existentes.

De manera más detallada, las reglas fundamentales de aplicación de estilos en caso de conflicto son las siguientes:

- La regla definida con el selector universal (*) se impone al resto de reglas definidas con selectores más específicos.
- Si una propiedad se modifica en un único punto, no hay conflicto y se aplica.
- Si una propiedad se modifica en el atributo **style**, se impone al intento de modificación de la misma propiedad en el elemento **<style>** o en una hoja CSS externa.
- Si una propiedad se modifica en el elemento **<style>**, se impone al intento de modificación de la misma propiedad en una hoja CSS externa.
- Si dos o más hojas CSS modifican la misma propiedad, se impone la hoja que se ha añadido en último lugar.
- Si dos reglas modifican la misma propiedad del mismo elemento, se impone aquella cuyo selector es más específico (a excepción del selector universal):
  - El selector con mayor número de **identificadores (id)** será el de mayor prioridad.
  - El selector con mayor número de **clases (class)** o pseudoclases será el siguiente en el orden de prioridad.
  - El selector con mayor número de elementos o pseudoelementos será el siguiente en el orden de prioridad.
- En caso de igualdad en las prioridades se impone el último estilo aplicado.

Estas reglas generales se pueden omitir o modificar mediante una serie de valores comunes a todas las propiedades que tienen efecto sobre cómo se realiza la herencia. Estos valores son:

- **inherit.** Activa la herencia, haciendo que la propiedad en la que se está asignando como valor herede la configuración del elemento padre.
- **initial.** Al indicar este valor a una propiedad se consigue que tome el valor de la hoja de estilos por defecto para el elemento al que hace referencia.
- **unset.** Restablece el valor de la propiedad a su valor natural.
- **revert.** Restablece el valor de la propiedad al valor que habría tenido si no hubiese sufrido cambios desde ningún origen.

Además, CSS dispone de la declaración **!important.** Este atributo anula las propiedades de un elemento como consecuencia de la aplicación de las reglas de asignación.

La declaración se efectúa inmediatamente después de la asignación del valor a la propiedad y junto antes del carácter ; que finaliza la asignación. En el siguiente código CSS se puede observar un ejemplo de aplicación:

```
div {
    color:rgb(9, 145, 224) !important;
}
```

Actividad propuesta 3.2

Prioridades

Dado el siguiente bloque de código HTML y estilo CSS, determina de qué color aparecerán estas cuatro citas del poeta italiano Dante Alighieri.

```
<p class="frases-celebres" id="frase1">
Aquél que escucha bien, toma apuntes.
</p>
<p id="frase2">
Si no se modera tu orgullo, él será tu mayor castigo
</p>
<p class="frases-celebres">
Una poderosa llama sigue a una pequeña chispa.
</p>
<p>
El secreto para que las cosas sean hechas está en hacerlas.
</p>

#frase1 {
    color: blue;
}

.frases-celebres {
    color: green;
}

p {
    color: red;
}
```

3.5. Valores y unidades en CSS

Antes de presentar las posibilidades de las hojas de estilo es necesario conocer qué valores admiten los diferentes parámetros y en qué unidades se expresan.

3.5.1. Tipos de datos

Los tipos de datos que se utilizan a la hora de configurar un estilo son los que se detallan en la Tabla 3.2.

Tabla 3.2. Tipos de datos admitidos por CSS

Tipo	Descripción
Entero	Número entero, ya sea positivo o negativo. Por ejemplo 10 o –3.
Número	Un número decimal. Utiliza el punto como separador decimal. Si no hay decimales, se omite el punto.

Tipo	Descripción
Dimensión	Es un número con una dimensión asociada; deg (grados), s (segundos) o px (píxeles) son algunos ejemplos.
Porcentaje	Una fracción de un total, normalmente referente a la dimensión del padre del elemento al que se está aplicando. Por ejemplo 60 % en un **font-size** representa dicho porcentaje del tamaño del texto del elemento padre. Si, en cambio, se aplica un porcentaje a la propiedad **width,** se aplica dicho porcentaje sobre el ancho del elemento contenedor.

3.5.2. Unidades de longitud absoluta

Las unidades de longitud absoluta representan longitudes sin importar en qué contexto se encuentre el componente al que hacen referencia.

Tabla 3.3. Unidades absolutas

Unidad	Nombre
px	Píxel
cm	Centímetro (96 px / 2,54)
mm	Milímetro
Q	Cuarto de milímetro
in	Pulgada (96 px)
pt	Puntos (1/72 de in)
pc	Picas (1/16 de in)

3.5.3. Unidades de longitud relativas

Las unidades de longitud relativas representan longitudes que tienen diferente valor en función del contexto en el que se encuentran.

Tabla 3.4. Unidades relativas

Unidad	Nombre
em	Tamaño de letra del elemento padre.
ex	Altura de la fuente del elemento.
ch	Ancho del carácter «0» de la fuente del elemento.
rem	Tamaño de letra del elemento raíz.

Unidad	Nombre
lh	Altura de la línea del elemento.
vw	1 % del ancho de la ventana gráfica.
vh	1 % del alto de la ventana gráfica.
vmin	1 % de la dimensión más pequeña de la ventana gráfica.
vmax	1 % de la dimensión más grande de la ventana gráfica.

De este conjunto de posibles unidades, las más utilizadas son **em** y **rem.** Las dimensiones absolutas son más sencillas de aplicar, pero los resultados dependen mucho de la resolución y de la densidad de píxeles de la pantalla, por lo que suele ser más eficaz utilizar unidades relativas.

Ejemplo:

Se muestran tres textos con diferentes tamaños de letra: en los dos primeros casos se utiliza **em** como unidad de medida mientras que en el tercer caso se asigna un tamaño porcentual. El tamaño asignado al id **saludoTriplePorcentual** (300 %) y al id **saludoTriple** (3 em) tienen el mismo efecto, ya que ambos multiplican por tres el tamaño del tipo de letra del elemento.

```
<div>
   Buenos días normal
   <div id="saludoTriplePorcentual">
      Buenos días porcentual
   </div>
</div>
<div id="saludoDoble">
     Buenos días doble
</div>
<div id="saludoTriple">
     Buenos días triple
</div>
```

```
#saludoDoble {
   font-size: 2em;
}
#saludoTriple {
  font-size: 3em;
}
#saludoTriplePorcentual {
   font-size: 300%;
}
```

Buenos días normal

Buenos días porcentual

Buenos días doble

Buenos días triple

Figura 3.3. *En CSS se pueden expresar los tamaños en varios tipos de unidades.*

3.5.4. Colores

Los colores se pueden utilizar para colorear textos, fondos de contenedores o bordes, por ejemplo. CSS permite expresar los colores con varias notaciones distintas:

- Por nombre. Indicando el nombre del color de la lista de la recomendación del W3C *(CSS Color Module Level 3)*. Así, el color rojo se asigna con el valor **red.** La lista completa se encuentra en el siguiente enlace: https://www.w3.org/TR/css-color-3/#html4.
- Por valor RGB (del inglés *Red, Green, Blue;* «rojo, verde, azul»). Indicando la composición de las tres componentes con valores en el rango 0-255 mediante el formato **rgb(*r,g,b*)**. Por ejemplo el color rojo se asigna con el valor **rgb(255,0,0).** Con la variante **rgba(*r,g,b,a*)** se añade un componente que indica la transparencia del color (lo que se denomina **canal alfa**) con un valor decimal comprendido entre 0 (completamente transparente) y 1 (completamente opaco).
- Por valor hexadecimal. Indicando el valor en hexadecimal de las componentes RGB mediante el formato ***#valorhexadecimal.*** Por ejemplo, el color rojo se asigna con el valor **#ff0000.**
- Por valor HSL (del inglés *Hue, Saturation, Lightness*; «matiz, saturación, luminosidad»). Indicado los valores correspondientes para cada componente (*Hue* entre 0° y 360°; *Saturation* entre 0 % y 100 %; *Lightness* entre 0 % y 100 %) en formato **hsl(*h, s, l*).** Por ejemplo, el color rojo se asigna con el valor **hsl(0, 100 %, 50 %).** Con la variante **hsla(*h, s, l, a*)** se añade un componente que indica la transparencia del color con un valor decimal comprendido entre 0 (completamente transparente) y 1 (completamente opaco).

A continuación, se muestran dos ejemplos de asignación de color sobre un mismo código HTML expresando dichos colores con diferentes notaciones.

Código HTML:

```
<div id="noticia">
El diario <cite>ABC</cite> redactó la increíble noticia de la siguiente
manera: "Un pie se descuelga por las escalerillas del módulo lunar posado en
el Mar de la Tranquilidad."
</div>
```

Notación por valor RGB:

```
#noticia {
   padding: 1em;
   background-color: rgb(200,200,200);
}
```

El diario *ABC* redactó la increíble noticia de la siguiente manera: "Un pie se descuelga por las escalerillas del módulo lunar posado en el Mar de la Tranquilidad."

Figura 3.4. *Mediante RGB se pueden referenciar hasta un total de 16.777.216 colores distintos.*

Notación por nombre:

```
#noticia {
   padding: 1em;
   background-color: wheat;
}
```

El diario *ABC* redactó la increíble noticia de la siguiente manera: "Un pie se descuelga por las escalerillas del módulo lunar posado en el Mar de la Tranquilidad."

Figura 3.5. *La notación por nombre permite recordar los colores más fácilmente.*

3.5.5. Imágenes

Las referencias a imágenes admiten dos tipos de valores: archivos o degradados.

Los archivos se referencian utilizando la función **url()**. Esta referencia puede ser local o a un recurso externo.

Ejemplo:

Se asigna una imagen de fondo a un elemento ** .** La imagen de fondo del elemento ** ** con identificador **huertos** se obtiene de un fichero alojado en la misma carpeta en la que se encuentra el documento CSS.

```
<div id="huertos">
   <p>Los huertos ecológicos son cada vez más populares gracias a la calidad
de sus productos.</p>
</div>
```

```
#huertos {
   padding: 1em;
   color:whitesmoke;
   font-weight: bolder;
   background-image: url("huerta.jpg");
}
```

Figura 3.6. *Las imágenes de fondo son un recurso gráfico muy llamativo.*

La misma imagen, podría estar alojada en un servidor web. En ese caso, la referencia en la hoja de estilos CSS tendrá el siguiente aspecto:

```
background-image: url("http://www.paraninfo.es/huerta.jpg");
```

Un degradado o gradiente de color técnicamente no tiene la consideración de color y no se puede aplicar como tal, sino que se considera una imagen, por lo que su utilización debe realizarse en el contexto adecuado. Para crear una imagen con un gradiente hay que utilizar la palabra **linear-gradient, radial-gradient, conic-gradient, repeating-linear-gradient** o **repeating-radial-gradient** e indicar los colores que utilizar, la dirección, el momento de la transición entre colores... Las posibilidades son casi infinitas y permiten a los diseñadores dar rienda suelta a su creatividad.

Ejemplo:

Se muestra el elemento ** ** con un gradiente de color de fondo.

```
<div id="noticia">
El diario <cite>La Vanguardia</cite> publicó la noticia en portada
concediéndole la importancia que se merecía: "Los habitantes de la Tierra, en
vela permanente, contemplan, a través de la Televisión, con asombro y emoción
como un hombre llamado Neil Armstrong baja por una escalera y pone el pie en
la Luna".
</div>
```

```
#noticia {
   padding: 1em;
   background: linear-gradient(to bottom, rgb(210,190,190), aquamarine);
}
```

El diario *La Vanguardia* publicó la noticia en portada concediéndole la importancia que se merecía: "Los habitantes de la Tierra, en vela permanente, contemplan, a través de la Televisión, con asombro y emoción como un hombre llamado Neil Armstrong baja por una escalera y pone el pie en la Luna".

Figura 3.7. *Los gradientes son altamente configurables.*

3.5.6. Tipos de letras

En computación se conoce como *tipo de letra, tipografía* o *fuente* al conjunto de modelos gráficos que representan cada uno de los caracteres y símbolos representables por el ordenador y que se almacena en un fichero. Existe un número casi ilimitado de tipos de letras creados por los diseñadores gráficos y, como puede imaginarse, no están todos disponibles en todos los ordenadores.

Para poder disponer de un tipo de letra específico hay que tener el fichero que almacena la información gráfica en el ordenador o, en caso contrario, algún mecanismo para obtener dicha información en el momento de su uso.

Desde el punto de vista del diseño web, existen ciertas tipografías que se pueden utilizar de manera segura y general, ya que son comunes a todos los sistemas. Se denominan **tipos de letra seguros** y la lista no es muy extensa: **Arial, Courier New, Georgia, Times New Roman, Trebuchet MS** y **Verdana.** De manera específica CSS define cinco tipos de letra genéricos cuya representación puede no ser exactamente la misma en todos los sistemas. Estos tipos genéricos son: **serif, sans-serif, monospace, cursive** y **fantasy.**

Este texto está escrito con tipo de letra elegido por defecto por el navegador

Este texto está escrito con tipo de letra Courier New

Este texto está escrito con tipo de letra monospace

Figura 3.8. *Solo algunas tipografías están disponibles en todos los ordenadores.*

A partir de este conjunto de opciones básicas de tipos de letras, se abre un universo de múltiples y muy atractivas tipografías creadas por diseñadores y artistas. De forma general, los navegadores admiten formatos de letra diferentes, por lo que hay que especificar alternativas para todos ellos. El formato más común entendido por la mayoría de los navegadores es *The Web Open Font Format* en sus versiones 1 y 2 (extensiones woff y woff2). Además, un gran número de navegadores admiten los tipos de letra *TrueType Fonts* (extensión .ttf), *OpenType Fonts* (extensión .otf) y *SVG Fonts* (extensión .svg).

Se puede referenciar una fuente no genérica en la hoja de estilos CSS. Si está instalada en el ordenador, se mostrará correctamente. Si no está disponible, se mostrará con el tipo de letra por defecto.

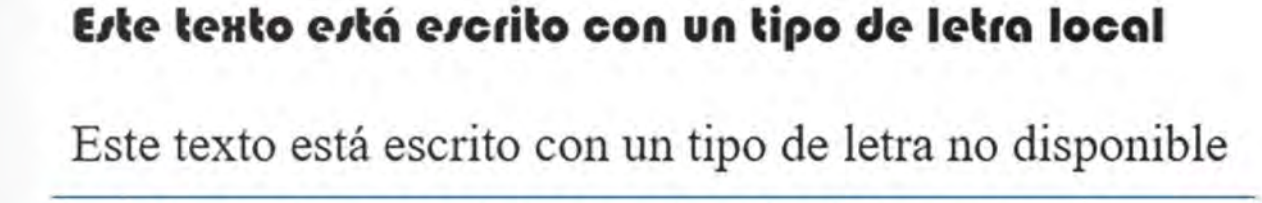

Figura 3.9. *Los navegadores tienen un plan alternativo si el tipo de letra elegido no está disponible.*

Para evitar los inconvenientes surgidos por no tener instalado el tipo de letra en los ordenadores de los usuarios, CSS permite cargar la tipografía desde un servidor.

La carga de tipografías se hace con la regla @font-face. Tiene la siguiente sintaxis básica:

```
@font-face {
        font-family: alias-de-la-fuente;
        src: url("url-de-la-fuente");
}
```

El siguiente código CSS asigna al alias **fuente-local** una fuente almacenada en el fichero **Rubik-Bold.ttf.** Este fichero estará almacenado en el servidor y lo descargará el cliente cuando sea necesario, garantizando la disponibilidad del tipo de letra.

```
@font-face {
    font-family: fuente-local;
    src: url("fonts/Rubik-Bold.ttf");
}
```

Por supuesto, el fichero, con la tipografía, puede estar almacenado en un servidor distinto del que aloja a la página HTML, tal y como se puede ver en el siguiente código de ejemplo:

```
@font-face {
    font-family: 'Hanalei Fill';
    src: url("https://fonts.gstatic.com/s/hanaleifill/v9/
fC1mPYtObGbfyQznIaQzPQi8UAjA.woff2");
}
```

Otra alternativa para importar fuentes de letra externas a la página web consiste en utilizar la regla **@import.** Esta regla permite obtener un recurso a partir de una URL.

En el siguiente código se muestra la regla de carga y el uso de una fuente de letra:

```
@import url('https://fonts.googleapis.com/css2?family=Roboto');

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

Debido a que el formato del fichero que contiene el tipo de letra puede no ser compatible con el navegador, es recomendable proporcionar alternativas que incluyan varios formatos distintos.

Nota técnica

No todas las fuentes son gratuitas. Asegúrate de que tienes los derechos adecuados sobre las fuentes que uses en tus páginas y aplicaciones web.

3.6. Selectores CSS

Los selectores constituyen el mecanismo que determina sobre qué elemento o elementos se debe aplicar un estilo. Es un sistema muy elaborado y capaz de conseguir un alto nivel de precisión a la hora de determinar sobre qué propiedades de qué elementos se van a aplicar los valores deseados.

Se puede considerar los selectores como una especie de cribas de diferentes calibres, capaces de permitir o impedir el paso de los elementos en función de su tamaño. En el caso concreto de CSS, el tamaño del elemento no es relevante, pero los elementos de HTML disponen de otros atributos que pueden participar de este proceso de selección; solo a aquellos elementos que «pasen el filtro» se les aplicará el estilo correspondiente.

Figura 3.10. *CSS aplica el estilo a un elemento si este encaja con el selector.*

Los selectores tienen un nivel de especificidad dispar, desde muy generales hasta muy específicos. Lo mismo ocurre con el nivel de complejidad; hay selectores muy sencillos y fáciles de comprender y otros más elaborados y complejos.

Se pueden agrupar los selectores como básicos, combinadores, pseudoclases y pseudoelementos.

3.6.1. Selectores básicos

Son los selectores más sencillos de entender y de uso más habitual. Se detallan en la Tabla 3.5.

Tabla 3.5. Selectores básicos

Nombre	Selecciona	Sintaxis
Universal	Todos los elementos.	*
De tipo	Todos los elementos de un tipo determinado.	*nombreElemento*
De clase	Todos los elementos de una determinada clase.	*.nombreClase*
De identificador	El elemento con el identificador indicado.	*#identificador*
De atributo	Los elementos que tienen un determinado atributo con unas características específicas. = → El atributo tiene un determinado valor. *= → El atributo contiene un determinado valor. ^= → El atributo comienza con un determinado valor. $= → El atributo termina con un determinado valor. ~= → El atributo contiene un determinado valor en una lista de valores.	[*atributo*] *elemento*[*atributo*] *elemento*[*atributo*="*cadena*"] *elemento*[*atributo**="*cadena*"] *elemento*[*atributo*^="*cadena*"]] *elemento*[*atributo*$="*cadena*"]] *elemento*[*atributo*~="*cadena*"]]

Ejemplo:

Aquí se utilizan selectores de tipo universal, de elemento, de clase, de identificador y de atributo.

```
<body>
<h1>RESUMEN ANUAL</h1>
<form action="">
   <input type="text" placeholder="Departamento">
   <input type="number" placeholder="Límite de ventas">
</form>
<p id="introduccion">
   Durante la temporada los resultados han sido satisfactorios...
</p>
<table>
   <tr>
      <th>Ingresos</th>
      <th>Gastos</th>
      <th>Diferencia</th>
   </tr>
  <tr>
     <td>100</td>
```

```
      <td>50</td>
      <td class="positivo">50</td>
   </tr>
   <tr>
      <td>200</td>
      <td>190</td>
      <td class="position">10</td>
   </tr>
   <tr>
      <td>80</td>
      <td>180</td>
      <td class="negativo">-100</td>
   </tr>
</table>
</body>
```

```
/* Selector universal */
* {
   margin:3px;
}

/* Selector de elemento */
h1 {
   color: rgb(50, 50, 50);
   font-style: italic;
}

/* Selector de identificador */
#introduccion {
   border-style: dotted;
   padding: 1em;
}

/* Selectores de clase */
.positivo {
   color:green;
   font-weight: bolder;
}
.negativo {
   color: red;
   font-weight: bolder;
}

/* Selectores de atributo */
input[type]{
   background-color: rgb(196, 248, 196);
}
input[type=text]{
   color: rgb(0, 98, 255);
   font-weight: bolder;
}
```

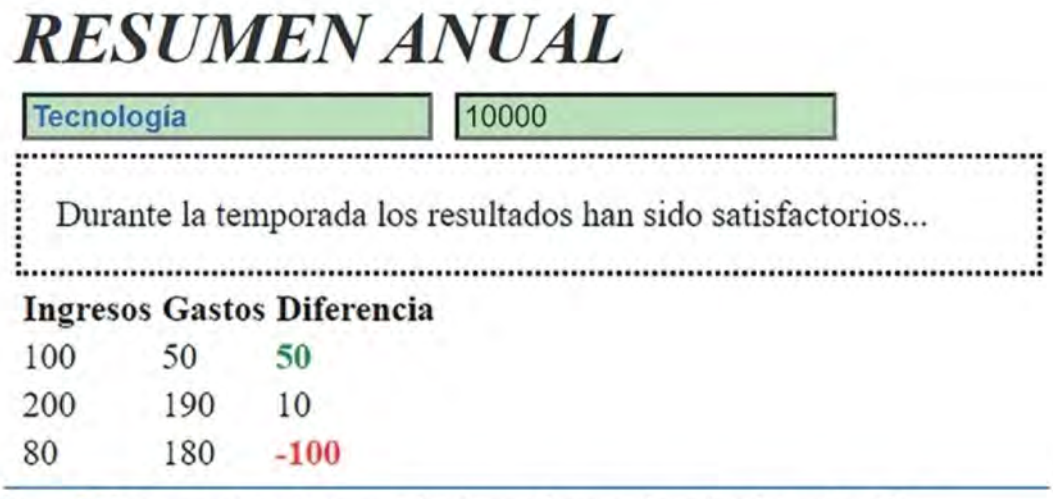

Figura 3.11. *Conocer los selectores es fundamental para poder aplicar los estilos CSS.*

3.6.2. Agrupación de selectores

En ocasiones se desea aplicar el mismo estilo a varios elementos. Una posible solución consiste en crear un selector distinto para cada uno de ellos con las mismas propiedades y los mismos valores, aunque esta solución puede no ser la mejor.

CSS permite crear reglas con selectores múltiples pudiendo así aplicar las mismas declaraciones a distintos grupos de elementos.

La sintaxis es:

```
elemento, elemento, elemento { propiedad : valor }
```

Ejemplo:

Creación de un selector que aplique el mismo estilo a varios elementos.

```
<div>
<h1>CSS</h1>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
<h2>Introducción</h2>
<p>Consequuntur veniam, quaerat repellendus facilis laborum magni
reprehenderit.</p>
<h2>Breve historia</h2>
<p>Doloribus fugit, nesciunt sequi pariatur adipisci rem nostrum optio...</p>
</div>
```

```
h1, h2, h3 {
color: blue;
}
```

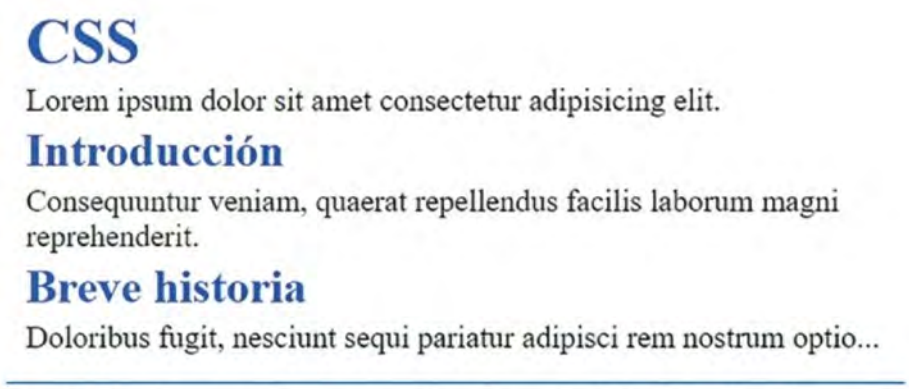

Figura 3.12. *Al agrupar selectores se reduce el tamaño de las hojas de estilo.*

3.6.3. Combinadores

Son selectores en los que se tiene en cuenta la relación entre elementos en la estructura jerárquica del documento.

Tabla 3.6. Combinadores

Nombre	Combinador	Descripción
De hermanos	A ~ B	A y B son hermanos.
De hijos	A > B	B es hijo de A.
De hermanos adyacentes	A + B	A y B son hermanos y B está inmediatamente a continuación de A.
De descendientes	A B	B es descendiente de A, pero no necesariamente es hijo directo.

Ejemplo:

Dado el siguiente código HTML:

```
<div>
   <div>
      <h2>Sistema operativo</h2>
      <p>El sistema operativo es el componente...</p>
   </div>
   <article>
      <p>Un buen antivirus protege al ordenador...</p>
   </article>
</div>
```

Esta regla centra el texto de los elementos **<p>** que sean hijos de un elemento **<article>**.

```
article > p {
   text-align: center;
}
```

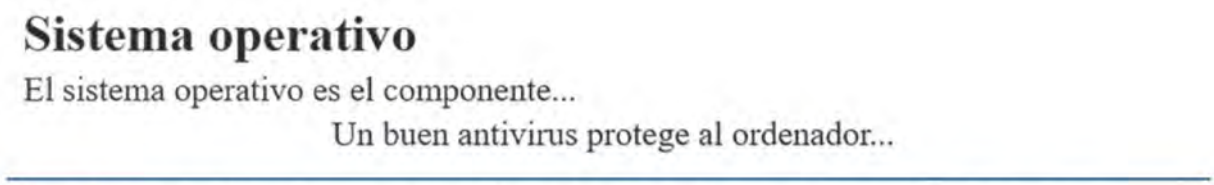

Figura 3.13. *Los combinadores son más complejos de usar, pero proporcionan mayor nivel de precisión.*

Esta regla centrará el texto de los elementos **<p>** que sean hermanos de un elemento **<h2>**.

```
h2 ~ p {
   text-align: center;
}
```

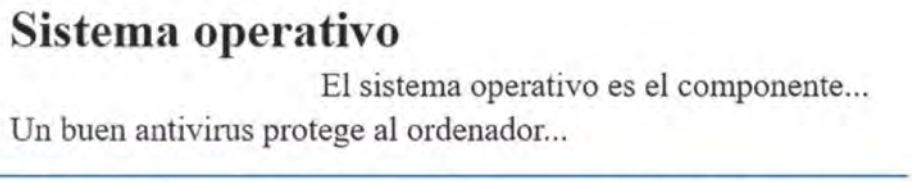

Figura 3.14. *Los selectores creados con combinadores posibilitan múltiples alternativas.*

Esta regla centrará el texto de los elementos **<p>** que sean descendientes de un elemento ** **.

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

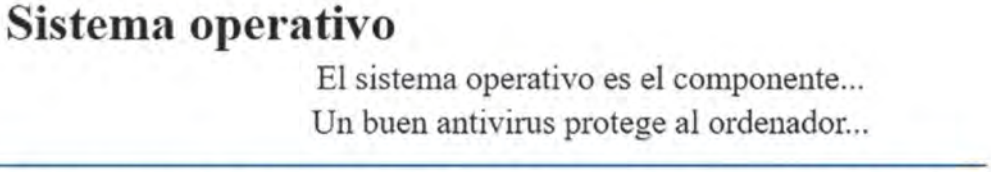

Figura 3.15. *Prácticamente todas las relaciones entre elementos se pueden expresar con combinadores.*

3.6.4. Pseudoclases

Se añade como palabra clave a continuación un selector convencional, creando un nuevo filtro de selección en función del estado del elemento o de los elementos filtrados por el selector.

La sintaxis es:

```
selector:pseudoclase { propiedad: valor; }
```

En la Tabla 3.7 se muestra una extensa relación de las pseudoclases existentes. Se omiten aquellas que son experimentales y no tienen un soporte generalizado por parte de los navegadores o cuyo uso es poco habitual.

Tabla 3.7. Pseudoclases

Nombre	Estado del elemento o elementos proporcionados por el selector
:active	El elemento ha sido activado por el usuario.
:checked	Afecta a elementos **<input>** de tipo **radio** o **checkbox** o a elementos **<option>** cuando han sido marcados.
:default	Elemento de un formulario marcado como predeterminado.
:disabled	El elemento está deshabilitado.
:empty	El elemento no tiene hijos.
:enabled	El elemento está habilitado.
:first-child	El primer elemento de entre un grupo de elementos hermanos.
:first-of-type	El primer elemento de un tipo de entre un grupo de elementos hermanos.
:focus	El elemento (de un formulario) tiene el foco.
:focus-within	El elemento (de un formulario) o uno de los elementos contenidos tiene el foco.
:hover	El cursor del ratón se encuentra sobre el elemento.
:indeterminate	El estado es indeterminado. Aplica a los elementos **<input>** de tipo **checkbox** y **radio** y a los elementos **<progress>.**
:in-range	El valor de un campo **<input>** se encuentra dentro del rango indicado por los atributos **min** y **max.**
:invalid	El elemento (de un formulario) es válido.
:lang(*idldioma*)	El elemento está en un idioma determinado. Por ejemplo **:lang(es)**
:last-child	El último elemento de un grupo de elementos hermanos.
:last-of-type	El último elemento de un tipo de entre un grupo de elementos hermanos.
:link	El enlace referenciado por el elemento no ha sido visitado.
:not()	Los elementos que no coinciden con una lista de selectores.
:nth-child(*n*)	El elemento en la posición *n* de un grupo de elementos hermanos. Se puede utilizar **odd** para indicar las posiciones pares y **even** para las posiciones pares en el parámetro **n.**
:nth-last-child(*n*)	El elemento en la posición *n* de un grupo de elementos hermanos, contando desde el final.
:nth-last-of-type(*n*)	El elemento de un tipo en la posición *n* de un grupo de elementos hermanos contando desde el final.
:nth-of-type(*n*)	El elemento de un tipo en la posición *n* de un grupo de elementos hermanos.
:only-child	El elemento no tiene elementos hermanos.
:only-of-type	El elemento no tiene elementos hermanos de un determinado tipo.

Nombre	Estado del elemento o elementos proporcionados por el selector
:optional	El elemento (de tipo **<input>, <select>** o **<textarea>**) no tiene el atributo **required.**
:out-of-range	El valor de un campo **<input>** se encuentra fuera del rango indicado por los atributos **min** y **max.**
:read-only	El elemento no es editable.
:read-write	El elemento es editable.
:required	El elemento es obligatorio.
:root	El elemento de más alto nivel del árbol que representa la estructura de la página.
:target	La zona (por ejemplo, un ** **), que es referenciada desde un enlace interno.
:valid	El elemento (de un formulario) no es válido.
:visited	El enlace referenciado por el elemento ha sido visitado.

Ejemplo:

Utilizando pseudoclases, se colorea el primer elemento de la lista en rojo, el último en azul y el elemento que está en la posición 3 en verde.

```
<ol>
   <li>Plataformas</li>
   <li>Simulador</li>
   <li>Deportes</li>
   <li>Estrategia</li>
   <li>Aventura gráfica</li>
</ol>
```

```
li {
   font-weight: bolder;
}

li:first-child {
   color:red;
}

li:nth-child(3) {
   color:green;
}

li:last-child {
   color: blue;
}
```

1. Plataformas
2. Simulador
3. Deportes
4. Estrategia
5. Aventura gráfica

Figura 3.16. *Las pseudoclases proporcionan reglas relacionadas con la posición.*

Actividad resuelta 3.1

Asignación de estilo CSS

Asignar estilo a los elementos de la tabla que se define en el código HTML proporcionado, prestando especial interés a los selectores utilizados. Los requisitos son los siguientes:

- Al encabezado y al pie se les asigna un color marrón de fondo, un color de texto blanco, y se convierte el texto a mayúsculas.
- A la última columna se le pone el texto en negrita.
- A las filas impares del cuerpo de la tabla se les asigna un color de fondo añil y un color de texto blanco.
- A las filas pares del cuerpo de la tabla se les asigna un color de fondo blanco y un color de texto añil.

```
<table>
   <thead>
      <tr>
      <td>Concepto</td>
      <td>Primer semestre</td>
      <td>Segundo semestre</td>
      <td>Total</td>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>Alimentación</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
         <td>Tecnología</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
         <td>Deporte</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
        <td>Calzado</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
        <td>Viajes</td>
```

```
        <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
         <td>Papelería</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
   </tbody>
   <tfoot>
      <tr>
         <td>Total</td>
         <td>0000</td><td>0000</td><td>0000</td>
      </tr>
   </tfoot>
</table>
```

Solución

```
thead>tr, tfoot>tr{
   background-color: brown;
   color: white;
   text-transform: uppercase;
}

td:last-child{
   font-weight: bolder;
}

tbody>tr:nth-child(odd) {
   background-color: rgb(28, 76, 150);
   color: white;
}

tbody>tr:nth-child(even) {
   background-color:white;
   color: rgb(28, 76, 150);
}
```

CONCEPTO	PRIMER SEMESTRE	SEGUNDO SEMESTRE	TOTAL
Alimentación	1500	1750	3250
Tecnología	1500	1750	3250
Deporte	1500	1750	3250
Calzado	1500	1750	3250
Viajes	1500	1750	3250
Papelería	1500	1750	3250
TOTAL	0000	0000	0000

Figura 3.17. *Con las pseudoclases se puede diseñar espectaculares tablas de datos.*

3.6.5. Pseudoelementos

Permiten seleccionar elementos por reglas que no forman parte de la estructura general del documento HTML. Al igual que las pseudoclases, se añaden a los selectores, pero no para crear un nuevo nivel de filtrado, sino para seleccionar partes más precisas del conjunto de elementos proporcionados por el selector. La separación entre el selector y el pseudoelemento se suele hacer con dos puntos dobles (::) para distinguirlos de las pseudoclases, pero los navegadores entienden también los dos puntos simples (:).

La sintaxis es:

```
selector::pseudo-elemento { propiedad: valor; }
```

En la Tabla 3.8 se muestra una tabla con los pseudoelementos de uso más extendido.

Tabla 3.8. Pseudoelementos

Nombre	Descripción
::after	Permite añadir contenido con el atributo **content** después del elemento seleccionado.
::before	Permite añadir contenido con el atributo **content** antes del elemento seleccionado.
::first-letter	Primera letra del primer bloque de un elemento seleccionado.
::first-line	Primera línea del primer bloque de un elemento seleccionado.
::selection	Permite modificar el estilo de la selección realizada con el ratón.

Ejemplo:

Utilizando pseudoelementos:

- Añadir el texto «SEGUIR LEYENDO...» inmediatamente después del elemento con **id** «**ElQuijote**».
- Asignar **3em** como tamaño de la primera letra del elemento con **id** «**ElQuijote**».

```
<p id="ElQuijote">
En un lugar de La Mancha, de cuyo nombre no quiero acordarme,
no ha mucho tiempo que vivía un hidalgo
</p>
```

```
#ElQuijote:after {
   content: "SEGUIR LEYENDO...";
}
#ElQuijote::first-letter {
  font-size: 3em;
}
```

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo... Es, pues, de saber que este sobredicho hidalgo, los ratos que estaba ocioso —que eran los más del año—, SEGUIR LEYENDO...

Figura 3.18. *Con los pseudoelementos el nivel de granularidad del selector es muy alto.*

Actividad propuesta 3.3

Selectores

Dado el bloque de código HTML siguiente, escribe los selectores que se indican a continuación:

- Selector de todos los elementos <h2>.
- Selector del primer elemento <h2>.
- Selector del último elemento <h2>.
- Selector del elemento con identificador #principal.
- Selector de los elementos <h2> que son descendientes del elemento con identificador #principal.
- Selector de los elementos de la clase .Articulo.
- Selector de los elementos <h2> que son descendientes de los elementos de la clase .Articulo.
- Selector del último hijo de los elementos <article> que son descendientes del elemento con identificador #secundario.

```
<h2>Título de la sección</h2>
<div id="principal">
    Texto div principal
    <article class="Articulo">
        <h2>Artículo 1</h2>
        <p>Texto del artículo 1</p>
    </article>
    <article class="Articulo">
        <h2>Artículo 2</h2>
        <p>Texto del artículo 2</p>
    </article>
</div>
<div id="secundario">
    Texto div secundario
    <article class="Articulo">
        <h2>Artículo 3</h2>
        <p>Texto del artículo 3</p>
    </article>
    <article class="Articulo">
        <h2>Artículo 4</h2>
        <p>Texto del artículo 4</p>
    </article>
</div>
```

3.7. Propiedades CSS

En CSS es posible modificar prácticamente cualquier aspecto del estilo de un elemento de HTML, ya que el conjunto de propiedades es muy extenso y completo.

En esta unidad se tratan aspectos relativos al funcionamiento del modelo de maquetación utilizado en CSS, conocido como «modelo de cajas». Además, se muestran algunas de las propiedades y técnicas más habituales.

Es conveniente recordar que, además de las propiedades específicas de cada elemento, existen propiedades comunes que tienen relación con la herencia y la aplicación en cascada de las reglas CSS.

Recuerda

Las propiedades **inherit, initial, unset** y **revert** son comunes y afectan a todos los elementos, permitiendo definir su relación con los elementos que los contienen.

3.7.1. Contenedores y posicionamiento: el modelo de cajas

La maquetación en CSS se basa en el denominado «modelo de cajas». Según este concepto, todos los elementos se encuentran ubicados en un contenedor rectangular denominado **caja.**

Estas cajas tienen un conjunto de propiedades básicas comunes que afectan directamente a su ubicación en la página y al espacio que ocupan:

- **Margen exterior.** Es el espacio vacío que se agrega al elemento y que lo separa del resto de elementos circundantes. Su dimensión se modifica con la propiedad **margin** y sus variantes **margin-top, margin-botton, margin-left** y **margin-right.**
- **Margen interior.** Es el espacio vacío que se agrega al elemento y separa del borde a su contenido. Su dimensión se modifica con la propiedad **padding** y sus variantes **padding-top, padding-bottom, padding-left** y **padding-right.**
- **Borde.** Es un espacio que puede tener distintos tipos de relleno y que se encuentra entre el margen exterior y el margen interior. Se configura con la propiedad **border** y ocupa espacio, por lo que aumenta el tamaño del elemento. Algunas propiedades para configurar los bordes son **border-radius, border-style, border-width** y **border-color.**
- **Contorno.** Es similar al borde, pero no ocupa espacio ya que se dibuja encima del elemento. Se configura con las propiedades **outline, outline-color, outline-width** y **outline-style.**
- **Ancho.** Representa el ancho del espacio del contenedor que puede contener elementos. Se modifica con la propiedad **width.**
- **Alto.** Representa el alto del espacio del contenedor que puede contener elementos. Se modifica con la propiedad **height.**

En la imagen de la Figura 3.19, proporcionada por el inspector del Google Chrome en las «Herramientas para desarrolladores», se puede observar un ejemplo de qué espacio ocupan en un elemento cada una de las partes que componen una caja.

Figura 3.19. *Las propiedades margin, border y padding afectan a distintos aspectos del espacio del elemento.*

La Figura 3.19 se basa en los datos del siguiente ejemplo: se aplica a un elemento ** ** vacío un estilo en el que se define un ancho porcentual del 50% del espacio disponible, un alto absoluto de 200 píxeles, un margen exterior absoluto de 0 píxeles, un margen interior absoluto de 10 píxeles y un borde de 5 píxeles de ancho.

```
<div id="divExterior">
    <div id="divInterior"></div>
</div>
```

```
#divInterior{
    background-color: rgb(255, 148, 25);
    width: 100%;
    height: 100%;
}

#divExterior {
    width: 50%;
    height: 200px;
    margin: 0px;
    padding:10px;
    border-width: 5px;

    border-style: solid;
    border-color:brown;
    background-color:rgb(250, 230, 120);
}
```

El resultado se puede observar en la Figura 3.20. Pese a que el margen indicado en el estilo no tiene grosor, se aprecia un margen entre el ** ** y el navegador: el margen del propio elemento **<body>**.

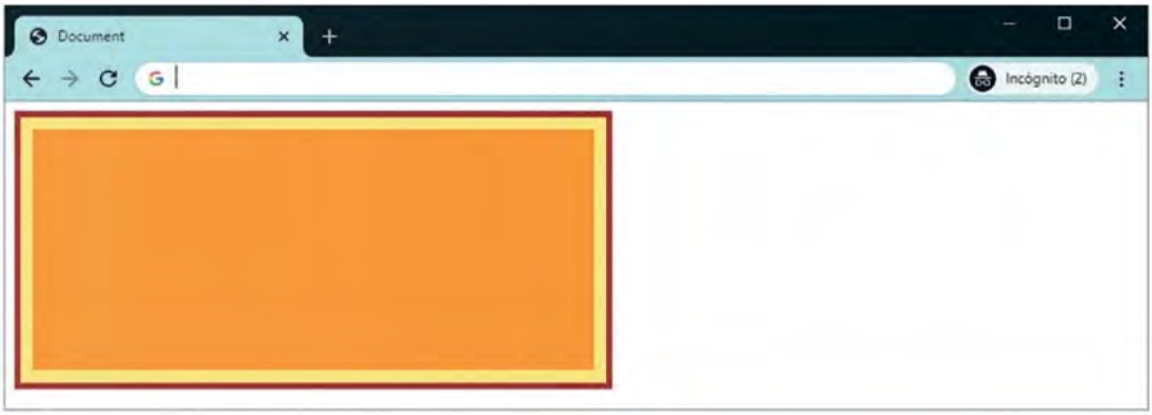

Figura 3.20. *El modelo de cajas proporciona control total sobre el espacio.*

Como se ha comentado anteriormente, los márgenes izquierdos y derechos se pueden asignar de manera independiente a cada uno de los lados:

```
#divExterior {
    width: 50%;
    height: 200px;
    margin: 0px;
    margin-left: 30px;
    padding:10px;
    padding-bottom: 50px;
    border-width: 5px;

    border-style: solid;
    border-color:brown;
    background-color:rgb(250, 230, 120);
}
```

Figura 3.21. *Border, padding y margin tienen variantes para determinar sobre qué lado de la caja actuar.*

Nota técnica

Para eliminar los márgenes del navegador y hacer que los contenidos aparezcan pegados a los bordes de este, se puede usar la siguiente regla:

```
body {
    margin:0px;
}
```

Hay dos tipos de cajas: **cajas en bloque** y **cajas en línea** (también llamadas de tipo *inline*). El que una caja sea de un tipo u otro tiene implicaciones directas en cómo se comporta la caja en relación con la página y con las demás cajas.

Las características fundamentales de las cajas en bloque son las siguientes:

- Después de la caja se realiza un salto de línea.
- Suele ocupar el 100% del espacio del contenedor en el que se encuentra.
- Se aplican los valores de las propiedades **width** y **height.**
- Los márgenes, bordes y rellenos desplazan al resto de cajas.

Por su parte, las características principales de las cajas en línea son estas:

- No generan salto de línea. Por defecto, ocupan el espacio que ocupa su contenido.
- No se aplican los valores de las propiedades **width** y **height.**
- Los márgenes, bordes y rellenos horizontales desplazan al resto de cajas en línea.
- Los márgenes, bordes y rellenos verticales no desplazan al resto de cajas en línea.

El siguiente código HTML contiene ejemplos de cajas en línea (elemento ****) y en bloque (elemento ** **).

```
<div>Caja de tipo bloque</div>
<div>Caja de tipo <span>en línea</span> dentro de una caja de tipo bloque</
div>
```

Como se puede observar en la Figura 3.22, el primer elemento ** ** provoca un salto de línea mientras que el elemento **** no lo hace.

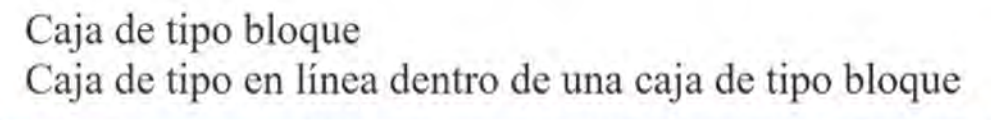

Figura 3.22. *Los elementos de bloque generan un salto de línea, los elementos de línea no.*

Todos los elementos de HTML pertenecen a uno de estos dos tipos por defecto, pero esta propiedad se puede modificar utilizando la propiedad **display,** asignándole los valores **inline** o **block** según se desee.

Por ejemplo, asignando al código HTML anterior una modificación de la propiedad **display,** el elemento **** comenzará a comportarse como un elemento de tipo caja.

```
span {
   display: block;
}
```

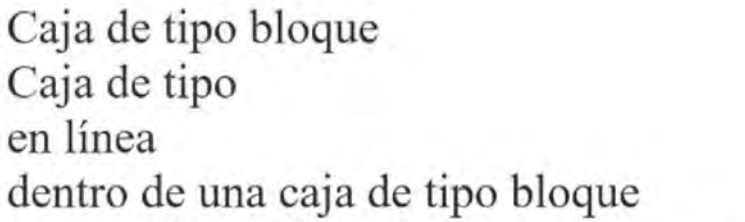

Figura 3.23. *Con la propiedad display, se puede modificar el comportamiento por defecto de los elementos.*

La relación de valores de la propiedad **display** es muy extensa, permitiendo incluso combinarlos. Se dividen en aquellos orientados a la visualización **interna** (cómo se distribuyen los componentes **dentro** de la caja) y los orientados a la visualización **externa** (cómo se comporta la caja en relación con los elementos que la rodean). Además, existe el valor especial **none** que desactiva todas las modificaciones de la propiedad **display** que afecten al elemento en cuestión y a todos los descendientes.

Los valores más utilizados para el tipo de visualización interna son los siguientes:

- **flex.** El elemento se comporta como un elemento de bloque y de acuerdo con el modelo denominado ***flexbox.*** Mediante la propiedad **flex-direction** se indica en qué dirección se desea que «fluya» el contenido del contenedor. A un contenedor con la propiedad **display** con valor **flex** se le denomina **contenedor flex.**
- **grid.** El elemento se comporta como un elemento de bloque y de acuerdo con el modelo de cuadrícula.

Ejemplos de aplicación del tipo de **display** interno **flex** con diferentes valores para la propiedad **flex-direction:**

```
<div id="divFlex">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
</div>
```

```
#divFlex {
    display: flex;
    flex-direction: row;
}
```

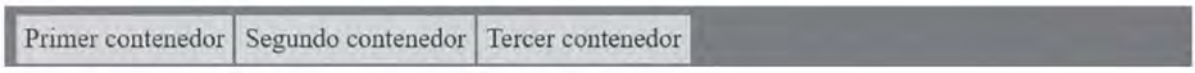

Figura 3.24. *El modelo flexbox facilita la alineación de los elementos.*

```
#divFlex {
    display: flex;
    flex-direction: row-reverse;
}
```

Figura 3.25. *Con flexbox, se puede invertir el orden natural de los elementos.*

```
#divFlex {
    display: flex;
    flex-direction: column;
}
```

Figura 3.26. *Flexbox trabaja en los ejes horizontal y vertical.*

```
#divFlex {
    display: flex;
    flex-direction: column-reverse;
}
```

Figura 3.27. *Flexbox proporciona la misma flexibilidad en vertical que en horizontal.*

Nota

El módulo de caja flexible o *flexbox* es un modelo de diseño que proporciona una gran versatilidad. En el siguiente enlace se puede obtener información detallada sobre este módulo de CSS3:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout

Ejemplos de aplicación del tipo de **display** interno **grid** con diferentes valores para la propiedad **grid-template-colums:**

```
<div id="divGrid">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
    <div>Cuarto contenedor</div>
    <div>Quinto contenedor</div>
    <div>Sexto contenedor</div>
</div>
```

```
#divGrid {
    display:grid;
    grid-template-columns: auto auto auto;
}
```

Primer contenedor	Segundo contenedor	Tercer contenedor
Cuarto contenedor	Quinto contenedor	Sexto contenedor

Figura 3.28. *El tipo de display grid distribuye los elementos como una tabla.*

```
#divGrid {
    display:grid;
    grid-template-columns: 100px 100px 100px;
}
```

Primer contenedor	Segundo contenedor	Tercer contenedor
Cuarto contenedor	Quinto contenedor	Sexto contenedor

Figura 3.29. *El tipo grid permite controlar el ancho de los elementos.*

```
#divGrid {
    display:grid;
    grid-template-columns: 70% 30%;
}
```

Primer contenedor	Segundo contenedor
Tercer contenedor	Cuarto contenedor
Quinto contenedor	Sexto contenedor

Figura 3.30. *La distribución de elementos por filas y columnas facilita la maquetación.*

Algunos de valores más utilizados para el tipo de visualización externa son los siguientes:

- **block.** El elemento se comporta como un elemento de bloque.
- **inline.** El elemento se comporta como un elemento en línea.

Referencia

Para profundizar en el conocimiento del parámetro **style** se pueden utilizar los siguientes enlaces:

W3C:

https://www.w3.org/TR/css-display-3/

MDN (Mozilla Development Network):

https://developer.mozilla.org/es/docs/Web/CSS/display

3.7.2. Propiedad float

Por defecto, los elementos de bloque tienen un comportamiento que genera grandes limitaciones a la hora de maquetar las páginas web. Por suerte, este comportamiento se puede modificar mediante los estilos CSS.

Los elementos de bloque que habitualmente se utilizan para maquetar son <header>, <footer>, <aside>, <nav>, <article>, <section> y , siendo este último el más genérico, ya que no tiene información semántica asociada. Es importante conocer el comportamiento por defecto de estos elementos y saber realizar las modificaciones adecuadas para que se distribuyan de la forma correcta.

Utilizando el elemento como referencia, en esta sección se muestran algunas técnicas relacionadas con la maquetación en las que se utiliza la propiedad **float** de los elementos de bloque.

Se puede ilustrar el comportamiento por defecto de los elementos de bloque con un sencillo ejemplo. Las siguientes reglas asignan colores distintos a tres identificadores:

```
.contenedor {
    color:white;
}
#contenedor1 {
    background-color: red;
}
#contenedor2 {
    background-color: green;
}
#contenedor3 {
    background-color: blue;
}
```

Los elementos de bloque, por defecto, ocupan todo el ancho del contenedor en el que se encuentran y se colocan unos debajo de otros, ya que insertan un salto de línea.

```
<div id="contenedor1" class="contenedor">Contenido del primer contenedor</div>
<div id="contenedor2" class="contenedor">Contenido del segundo contenedor</div>
<div id="contenedor3" class="contenedor">Contenido del tercer contenedor</div>
```

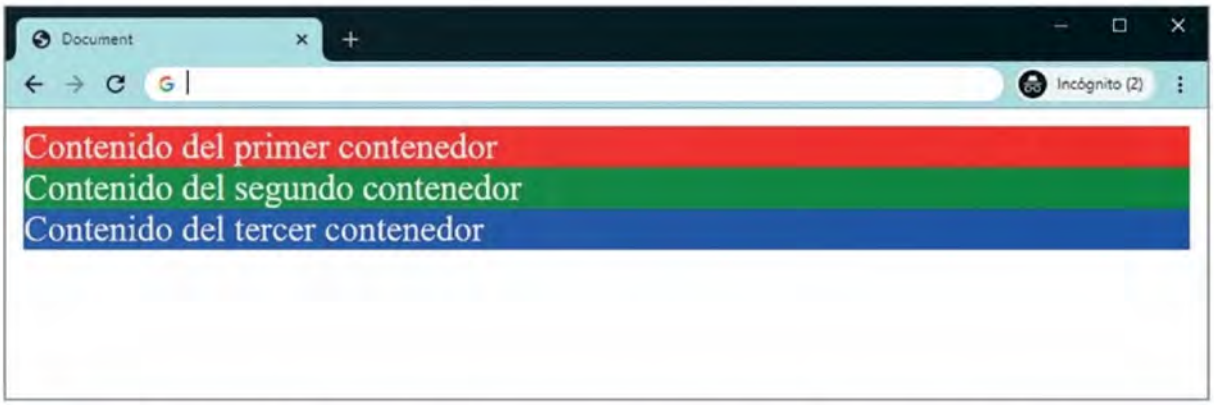

Figura 3.31. *Por defecto, los elementos de bloque ocupan todo el ancho del espacio disponible.*

Si se les asigna un ancho, ya sea en valores absolutos o relativos, los elementos de bloque siguen colocándose de forma vertical.

```
.contenedor {
    color:white;
    width: 30%;
}
```

Figura 3.32. *Por defecto, los elementos de bloque se posicionan uno debajo de otro, según el orden de aparición en el documento HTML.*

Este comportamiento por defecto se puede modificar mediante la propiedad **float**. Esta propiedad posiciona el elemento al lado izquierdo o derecho del contenedor en el que se encuentra. Los principales valores que admite son:

- **left:** «empuja» el elemento hacia el lado izquierdo de su contenedor.
- **right:** «empuja» el elemento hacia el lado derecho de su contenedor.
- **none:** no se «empuja» el elemento. Es el valor por defecto.
- **inherit:** hereda el valor de la propiedad **float** del contenedor padre.

Por ejemplo, modificando el estilo se puede forzar a que el elemento **contenedor1** se desplace hacia la izquierda hasta el límite de su contenedor, el elemento **contenedor2** se desplace hacia la derecha hasta el límite de su contenedor y el elemento **contenedor3** se desplace hacia la derecha hasta encontrarse con el elemento **contenedor2.**

```
#contenedor1 {
    background-color: red;
    float:left;
}
#contenedor2 {
    background-color: green;
    float:right;
}
#contenedor3 {
    background-color: blue;
    float:right;
}
```

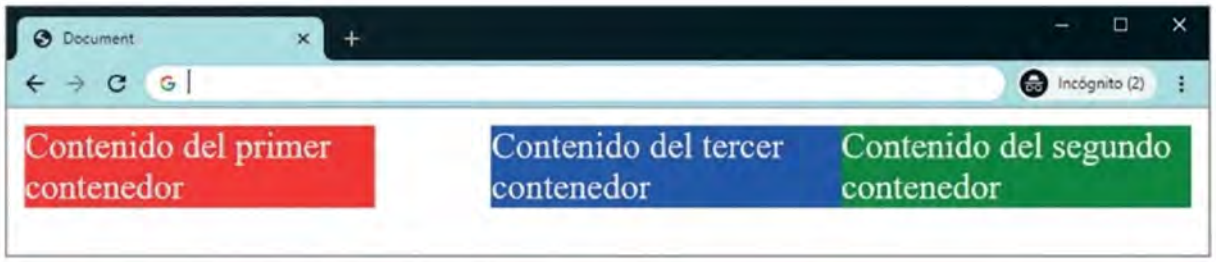

Figura 3.33. *Con la propiedad float se puede cambiar el comportamiento de los elementos de bloque.*

Si se modifica la regla aplicada a **contenedor3** dando a la propiedad **float** el valor **none**, dicho contenedor no flotará y se posicionará en su ubicación natural:

```
#contenedor3 {
    background-color: blue;
    float:none;
}
```

Nota técnica

Se puede centrar un elemento de tipo bloque fijándole un ancho y estableciendo **auto** como valor del atributo **margin.**

Figura 3.34. *El valor none de la propiedad float indica que el elemento no debe flotar.*

3.7.3. Propiedad position

La propiedad **position** determina cómo se posiciona un elemento en el documento. Es una propiedad muy útil para la maquetación, ya que proporciona una gran flexibilidad.

Los valores que admite esta propiedad se muestran en la Tabla 3.9.

Tabla 3.9. Valores admitidos por la propiedad position

Valor	Descripción
static	Es el valor por defecto. El elemento se posiciona siguiendo el flujo normal del documento. No le afectan las propiedades **top, bottom, left, right** y **z-index.**
relative	El elemento se posiciona siguiendo el flujo normal del documento, pero se puede desplazar de forma relativa a su posición por defecto utilizando las propiedades **top, bottom, left** y **right.** El desplazamiento no desplaza a los demás elementos, por lo que puede proporcionar apilamiento que deberá organizarse utilizando la propiedad **z-index.**
absolute	El elemento deja de seguir el flujo normal del documento. Mediante las propiedades **top, bottom, left** y **right** el elemento se posiciona de manera relativa a su ancestro. Si no tiene ancestro, se posiciona de manera relativa a la página completa.
fixed	El elemento deja de seguir el flujo normal del documento. Mediante las propiedades **top, bottom, left** y **right** el elemento se posiciona de manera relativa a la página completa.
sticky	El elemento se posiciona siguiendo el flujo normal del documento. Mediante las propiedades **top, bottom, left** y **right** se fijan unas posiciones límites relativas a su contenedor, de tal manera que no excederá de estas.

Ejemplo:

A continuación se hace uso del posicionamiento relativo para centrar un dentro de otro.

El principal (el que tiene el atributo **id** con valor «div1») ocupa todo el ancho disponible y tiene fijada una altura fija en píxeles. El tamaño y la posición del interior (el que tiene el atributo **id** con valor «div2») se declara relativo al elemento dentro del que se encuentra: el ancho es el 50 % del espacio de su contenedor; la posición es de 10

píxeles desde la parte superior del contenedor y de un 25 % del ancho del contenedor desde el lado izquierdo.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="position-relative.css">
    <title>Document</title>
</head>
<body>
    <div id="div1">
        <div id="div2">Texto del div con posición relativa</div>
    </div>
</body>
</html>
```

```
#div1 {
    width: 100%;
    height: 200px;
    background-color: aquamarine;

    position: static;
}
#div2 {
    width: 50%;
    height: 180px;
    background-color:lightgray;

    position: relative;
    top:10px;
    left:25%;
}
```

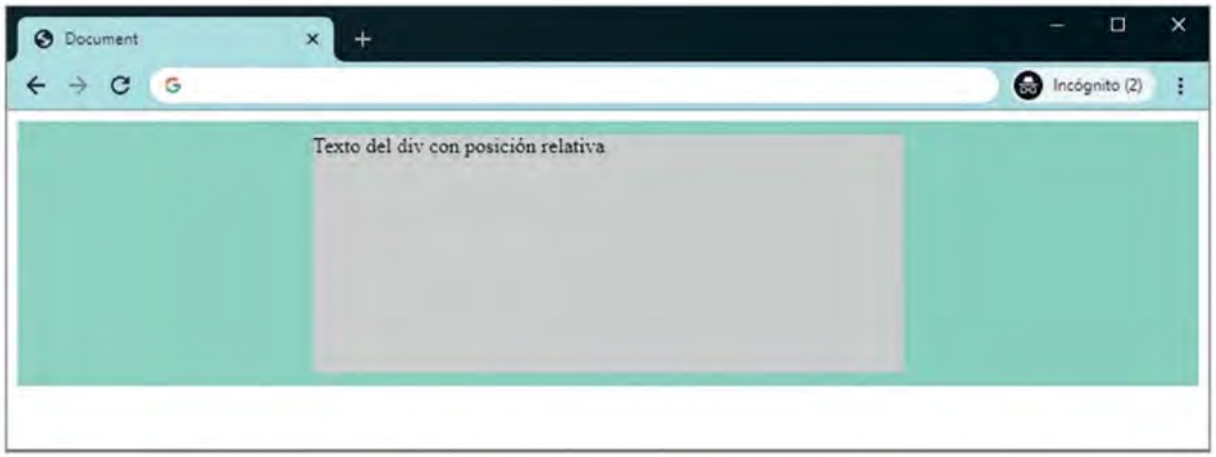

Figura 3.35. *Con el posicionamiento relativo los elementos se posicionan en función de su contenedor.*

Actividad resuelta 3.2

Creación de un pie de página fijo

Crear las reglas CSS para realizar, dado el código HTML que se muestra a continuación, un pie de página que permanezca fijo en la parte inferior del navegador, aunque se realice un desplazamiento *(scroll)* vertical sobre la página.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="position-fixed.css">
    <title>Posicionamiento fijo</title>
</head>
<body>
    <div id="div1">
        <!-- CONTENIDO -->
    </div>
    <div id="div2">
        <!-- CONTENIDO -->
    </div>
    <footer id="pie-documento">
        Este es el contenido del pie de la página
    </footer>
</body>
</html>
```

Solución

El elemento <footer> debe tener posicionamiento de tipo **fixed** y estar a 0 píxeles de la parte inferior del navegador (**bottom:0px**). El resultado es un contenedor que está «pegado» al extremo inferior de la página independientemente de la posición del resto de elementos.

```
body {
    margin:0px;
}
#div1 {
    width: 100%;
    background-color: rgb(255, 217, 0);
}
#div2 {
    width: 100%;
    background-color:rgb(0, 255, 255);
}
#pie-documento {
    width: 100%;
    height: 100px;
    background-color: rgb(150,150,150);
    position: fixed;
    bottom: 0px;
}
```