CURSO DE CSS2/3

POR
IVÁN RODRÍGUEZ RUIZ

ÍNDICE

1 **INTRODUCCIÓN**
- 1.1 ¿Qué es CSS?
- 1.2 Breve historia de CSS
- 1.3 Soporte de CSS en los navegadores
- 1.4 Especificación oficial
- 1.5 Funcionamiento básico de CSS
- 1.6 Cómo incluir CSS en un documento XHTML
- 1.7 Glosario básico
- 1.8 Medios CSS
- 1.9 Comentarios
- 1.10 Sintaxis de la definición de cada propiedad CSS

2 **SELECTORES**
- 2.1 Selectores básicos
- 2.2 Selectores avanzados
- 2.3 Agrupación de reglas
- 2.4 Herencia
- 2.5 Colisiones de estilos

3 **UNIDADES DE MEDIDA Y COLORES**
- 3.1 Unidades de medida
- 3.2 Colores

4 **MODELO DE CAJAS**
- 4.1 Anchura y altura
- 4.2 Margen y relleno
- 4.3 Bordes
- 4.4 Margen, relleno, bordes y modelo de cajas
- 4.5 Fondos

5 **POSICIONAMIENTO Y VISUALIZACIÓN**
- 5.1 Tipos de elementos
- 5.2 Posicionamiento
- 5.3 Posicionamiento normal
- 5.4 Posicionamiento relativo
- 5.5 Posicionamiento absoluto
- 5.6 Posicionamiento fijo
- 5.7 Posicionamiento flotante
- 5.8 Visualización

6 **TEXTO**
- 6.1 Tipografía
- 6.2 Texto

7 **ENLACES**
- 7.1 Estilos básicos
- 7.2 Estilos avanzados

8 **IMÁGENES**
- 8.1 Estilos básicos
- 8.2 Estilos avanzados

9 **LISTAS**
9.1 Estilos básicos
9.2 Estilos avanzados

10 **TABLAS**
10.1 Estilos básicos
10.2 Estilos avanzados

11 **FORMULARIOS**
11.1 Estilos básicos
11.2 Estilos avanzados

12 **LAYOUT**
12.1 Centrar una página horizontalmente
12.2 Centrar una página verticalmente
12.3 Estructura o layout
12.4 Alturas/anchuras máximas y mínimas
12.5 Estilos avanzados

13 **OTROS**
13.1 Propiedades shorthand
13.2 Versión para imprimir
13.3 Personalizar el cursor
13.4 Hacks y filtros
13.5 Prioridad de las declaraciones CSS
13.6 Validador
13.7 Recomendaciones generales sobre CSS

14 **RECURSOS ÚTILES**
14.1 Extensiones de Firefox
14.2 Aplicaciones web
14.3 Sitios web de inspiración
14.4 Referencias y colecciones de recursos

15 **NOVEDADES EN CSS3 (PARTE I)**
15.1 Efectos sobre Cajas y Textos
15.1.1 Border Radius
15.1.2 Sombras
15.1.2.1 La propiedad Box-Shadow
15.1.2.2 La Propiedad Text-Shadow
15.1.3 @font-face
15.1.4 Gradientes
15.2 Colores
15.2.1 RGBA
15.2.2 HSLA
15.3 Bordes
15.3.1 Outline
15.3.2 Border-image.
15.4 Transformaciones y Transiciones.
15.4.1 Transform: scale
15.4.2 Transform: rotate
15.4.3 Transform: skew
15.4.4 Transform: translate
15.4.5 El Shorthand Transform
15.4.6 Transformación Dinámica

15.5 Transiciones
15.5.1 Transition-property
15.5.2 Transition-duration.
15.5.3 Transition-timing-function
15.5.4 Transition-delay
15.5.5 Shorthand Transition
15.5.6 Propiedades que admiten transiciones Css

16 **NOVEDADES EN CSS3 (PARTE II)**
16.1 Introducción
16.2 Texto
16.2.1 text-align-last
16.2.2 word-break
16.2.3 word-wrap
16.2.4 Hyphens
16.2.5 Text-Overflow
16.2.6 Text-Stroke
16.2.7 ::selection
16.2.8 Columnas.
16.2.9 Font-size-adjust
16.2.10 font-stretch
16.3 Animaciones y máscaras
16.3.1 Marquesinas
16.3.2 Animaciones
16.3.3 Transformaciones 3D
16.3.4 Máscaras
16.3.5 Blend-Mode.
16.3.6 Eventos de Puntero
16.4 Fondos e Imágenes
16.4.1 Múltiples Fondos.
16.4.2 Background-Repeat.
16.4.3 Valor Local.
16.4.4 background-clip.
16.4.5 background-origin
16.4.6 background-size
16.4.7 La propiedad Box-Reflect.
16.4.8 CurrentColor
16.4.9 Filtros
16.5 Valores y Directivas
16.6 Flexbox
16.6.1 Propiedad Display Box
16.6.2 Estructura con Cajas Flexibles
16.6.3 Como centrar un elemento con flexbox
16.6.4 La propiedad Box-Sizing
16.6.5 Calc ()
16.7 Otras propiedades

1 INTRODUCCIÓN

1.1 ¿Qué es CSS?

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

1.2 Breve historia de CSS

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML, alrededor del año 1970. Desde la creación de SGML, se observó la necesidad de definir un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los documentos electrónicos.

El gran impulso de los lenguajes de hojas de estilos se produjo con el boom de Internet y el crecimiento exponencial del lenguaje HTML para la creación de documentos electrónicos. La guerra de navegadores y la falta de un estándar para la definición de los estilos dificultaban la creación de documentos con la misma apariencia en diferentes navegadores.

El organismo W3C (World Wide Web Consortium), encargado de crear todos los estándares relacionados con la web, propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML y se presentaron nueve propuestas. Las dos propuestas que se tuvieron en cuenta fueron la CHSS (Cascading HTML Style Sheets) y la SSP (Stream-based Style Sheet Proposal).

La propuesta CHSS fue realizada por Håkon Wium Lie y SSP fue propuesto por Bert Bos. Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS (Cascading Style Sheets).

En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS y lo añadió a su grupo de trabajo de HTML. A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como "CSS nivel 1".

A principios de 1997, el W3C decide separar los trabajos del grupo de HTML en tres secciones: el grupo de trabajo de HTML, el grupo de trabajo de DOM y el grupo de trabajo de CSS.

El 12 de Mayo de 1998, el grupo de trabajo de CSS publica su segunda recomendación oficial, conocida como "CSS nivel 2". La versión de CSS que utilizan todos los navegadores de hoy en día es CSS 2.1, una revisión de CSS 2 que aún se está elaborando (la última actualización es del 8 de septiembre de 2009). Al mismo tiempo, la siguiente recomendación de CSS, conocida como "CSS nivel 3", continúa en desarrollo desde 1998 y hasta el momento sólo se han publicado borradores.

1.3 Soporte de CSS en los navegadores

El trabajo del diseñador web siempre está limitado por las posibilidades de los navegadores que utilizan los usuarios para acceder a sus páginas. Por este motivo es imprescindible conocer el soporte de CSS en cada uno de los navegadores más utilizados del mercado.

Internamente los navegadores están divididos en varios componentes. La parte del navegador que se encarga de interpretar el código HTML y CSS para mostrar las páginas se denomina motor. Desde el punto de vista del diseñador CSS, la versión de un motor es mucho más importante que la versión del propio navegador.

La siguiente tabla muestra el soporte de CSS 1, CSS 2.1 y CSS 3 de los cinco navegadores más utilizados por los usuarios:

Navegador	Motor	CSS 1	CSS 2.1	CSS 3
Google Chrome	WebKit	Completo desde la versión 85 del motor	Completo	Todos los selectores, pseudo-clases y muchas propiedades
Internet Explorer	Trident	Completo desde la versión 7.0 del navegador	Completo	Todos los selectores, pseudo-clases y muchas propiedades a partir de la versión 10.0 del navegador
Firefox	Gecko	Completo desde la versión 1.0 del navegador	Completo	Todos los selectores, pseudo-clases y muchas propiedades
Safari	WebKit	Completo desde la versión 85 del motor	Completo	Todos los selectores, pseudo-clases y muchas propiedades
Opera	Presto	Completo desde la versión 1.0	Completo	Todos los selectores, pseudo-clases y muchas propiedades

Los navegadores Firefox, Chrome, Safari y Opera son los más avanzados en el soporte de CSS, ya que incluyen muchos elementos de la futura versión CSS 3 y un soporte casi perfecto de la actual version 2.1.

Por su parte, el navegador Internet Explorer sólo puede considerarse adecuado desde el punto de vista de CSS a partir de su versión 7. Internet Explorer 6, utilizado todavía por un número no despreciable de usuarios, sufre carencias muy importantes y contiene decenas de errores en su soporte de CSS. Internet Explorer 8 soporta casi todas las propiedades y características de CSS 2.1.

La tabla anterior ha sido elaborada a partir de la información que se puede encontrar en la página Comparison of layout engines de la Wikipedia, donde se muestra una comparación exhaustiva sobre el soporte de todas las características de CSS por parte de cada navegador.

1.4 Especificación oficial

La especificación o norma oficial que se utiliza actualmente para diseñar páginas web con CSS es la versión CSS 2.1, actualizada por última vez el 7 de junio de 2011 y que se puede consultar libremente en w3.org/TR/CSS21

Desde hace varios años, el organismo W3C trabaja en la elaboración de la próxima versión de CSS, conocida como CSS 3. Esta nueva versión incluye multitud de cambios importantes en todos los niveles y es mucho más avanzada y compleja que CSS 2. Puedes consultar el estado actual de cada componente de CSS 3 en w3.org/Style/CSS/current-work. También existe un blog oficial en el que se publican todas las novedades relacionadas con el estándar CSS.

1.5 Funcionamiento básico de CSS

Antes de que se generalizara el uso de CSS, los diseñadores de páginas web utilizaban etiquetas HTML especiales para modificar el aspecto de los elementos de la página. El siguiente ejemplo muestra una página HTML con estilos definidos sin utilizar CSS.
Nos creamos una carpeta en htdocs llamada ejemplosCSS.

NOTA IMPORTANTE: Como se puede comprobar he elegido el formato de codificación de HTML5 frente al XHTML porque simplifica el código, además de mejorar la visualización de los ejemplos.

Y aquí el primer ejemplo (que no lleva CSS) **Ej01.05a-SinCSS.html)**:

```
<!DOCTYPE html>
<html lang="es">
<head>
   <title>Formularios</title>
   <meta charset="UTF-8" />
</head>
<body>
   <section>
      <form action="maneja_formularioHTML5.php" method="post" autocomplete="yes">
         <p> Formulario: </p>
         <input pattern="[0-9]{5}" name="cp" title="Inserte Código Postal" /><br />
         <input type="submit" value="Enviar" />
      </form>
   </section>
</body>
</html>
```

El ejemplo anterior utiliza la etiqueta <font> con sus atributos color, face y size para definir el color, el tipo y el tamaño de letra de cada elemento de la página.

El problema de utilizar este método para definir el aspecto de los elementos se puede ver claramente con el siguiente ejemplo: si la página tuviera 50 elementos diferentes, habría que insertar 50 etiquetas <font>. Si el sitio web entero se compone de 10.000 páginas diferentes, habría que definir 500.000 etiquetas <font>. Como cada etiqueta <font> tiene tres atributos, habría que definir 1.5 millones de atributos.

Como el diseño de los sitios web está en constante evolución, es habitual modificar cada cierto tiempo el aspecto de las páginas del sitio. Siguiendo con el ejemplo anterior, cambiar el aspecto del sitio requeriría modificar 500.000 etiquetas y 1.5 millones de atributos.

La solución que propone CSS es mucho mejor; un ejemplo (**Ej01.05b-ConCSS.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
    h1 { color: red;  font-family: Arial;   font-size: large;  }
    p  { color: gray; font-family: Verdana; font-size: medium; }
  </style>
</head>
<body>
    <section>
      <h1>Titular de la página</h1>
      <p>Un párrafo de texto no muy largo.</p>
    </section>
</body>
</html>
```

CSS permite separar los contenidos de la página y la información sobre su aspecto. En el ejemplo anterior, dentro de la propia página HTML se crea una zona especial en la que se incluye toda la información relacionada con los estilos de la página.

Utilizando CSS, se pueden establecer los mismos estilos con menos esfuerzo y sin ensuciar el código HTML de los contenidos con etiquetas <font>. Como se verá después, la etiqueta <style> crea una zona especial donde se incluyen todas las reglas CSS que se aplican en la página.

En el ejemplo anterior, dentro de la zona de CSS se indica que todas las etiquetas <h1> de la página se deben ver de color rojo, con un tipo de letra Arial y con un tamaño de letra grande. Además, las etiquetas <p> de la página se deben ver de color gris, con un tipo de letra Verdana y con un tamaño de letra medio.

Definir los estilos de esta forma ahorra miles de etiquetas y millones de atributos respecto a la solución anterior, pero sigue sin ser una solución ideal. Como los estilos CSS sólo se aplican en la página que los incluye, si queremos que las 10.000 páginas diferentes del sitio tengan el mismo aspecto, se deberían copiar 10.000 veces esas mismas reglas CSS. Más adelante se explica la solución que propone CSS para evitar este problema.

1.6 Cómo incluir CSS en un documento XHTML

Una de las principales características de CSS es su flexibilidad y las diferentes opciones que ofrece para realizar una misma tarea. De hecho, existen tres opciones para incluir CSS en un documento HTML.

1.6.1 Incluir CSS en el mismo documento HTML

Los estilos se definen en una zona específica del propio documento HTML. Se emplea la etiqueta <style> de HTML y solamente se pueden incluir en la cabecera del documento (sólo dentro de la sección <head>). Un ejemplo (**Ej01.06a-CSSHead.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
    h1 { color: red;  font-family: Arial;   font-size: large;  }
    p  { color: gray; font-family: Verdana; font-size: medium; }
  </style>
</head>
<body>
    <section>
      <h1>Titular de la página</h1>
      <p>Un párrafo de texto no muy largo.</p>
    </section>
</body>
</html>
```

Este método se emplea cuando se define un número pequeño de estilos o cuando se quieren incluir estilos específicos en una determinada página HTML que completen los estilos que se incluyen por defecto en todas las páginas del sitio web.

El principal inconveniente es que si se quiere hacer una modificación en los estilos definidos, es necesario modificar todas las páginas que incluyen el estilo que se va a modificar.

NOTA: Los ejemplos mostrados en estos apuntes utilizan este método para aplicar CSS al contenido HTML de las páginas. De esta forma el código de los ejemplos es más conciso y se aprovecha mejor el espacio. Aunque evidentemente no es el mas recomendable.

1.6.2 Definir CSS en un archivo externo

En este caso, todos los estilos CSS se incluyen en un archivo de tipo CSS que las páginas HTML enlazan mediante la etiqueta <link>. Un archivo de tipo CSS no es más que un archivo simple de texto cuya extensión es .css Se pueden crear todos los archivos CSS que sean necesarios y cada página HTML puede enlazar tantos archivos CSS como necesite.

Si se quieren incluir los estilos del ejemplo anterior en un archivo CSS externo, se deben seguir los siguientes pasos:

1. Se crea un archivo de texto (en la carpeta css dentro de ejemplosCSS) llamado estilos.css y se le añade solamente el siguiente contenido:
   `p { color: blue; font-family: Verdana; }`
2. Se guarda el archivo de texto con el nombre estilos.css Se debe poner especial atención a que el archivo tenga extensión .css y no .txt
3. En la página HTML se enlaza el archivo CSS externo mediante la etiqueta <link>

Un ejemplo (**Ej01.06b-CSSExterno.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <link rel="stylesheet" type="text/css"
   href="css/Ej01.06b-CSSExterno.css" media="screen" />
</head>
<body>
    <section>
      <p>Un párrafo de texto.</p>
    </section>
</body>
</html>
```

Cuando el navegador carga la página HTML anterior, antes de mostrar sus contenidos también descarga los archivos CSS externos enlazados mediante la etiqueta <link> y aplica los estilos a los contenidos de la página.

Normalmente, la etiqueta <link> incluye cuatro atributos cuando enlaza un archivo CSS:

- rel: indica el tipo de relación que existe entre el recurso enlazado (en este caso, el archivo CSS) y la página HTML. Para los archivos CSS, siempre se utiliza el valor stylesheet
- type: indica el tipo de recurso enlazado. Sus valores están estandarizados y para los archivos CSS su valor siempre es text/css
- href: indica la URL del archivo CSS que contiene los estilos. La URL indicada puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.
- media: indica el medio en el que se van a aplicar los estilos del archivo CSS. Más adelante se explican en detalle los medios CSS y su funcionamiento.

De todas las formas de incluir CSS en las páginas HTML, esta es la más utilizada con mucha diferencia. La principal ventaja es que se puede incluir un mismo archivo CSS en multitud de páginas HTML, por lo que se garantiza la aplicación homogénea de los mismos estilos a todas las páginas que forman un sitio web.

Con este método, el mantenimiento del sitio web se simplifica al máximo, ya que un solo cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todas las páginas HTML que enlazan ese archivo.

Aunque generalmente se emplea la etiqueta <link> para enlazar los archivos CSS externos, también se puede utilizar la etiqueta <style>. La forma alternativa de incluir un archivo CSS externo se muestra a continuación:

De todas las formas de incluir CSS en las páginas HTML, esta es la más utilizada con mucha diferencia. La principal ventaja es que se puede incluir un mismo archivo CSS en multitud de páginas HTML, por lo que se garantiza la aplicación homogénea de los mismos estilos a todas las páginas que forman un sitio web.

Con este método, el mantenimiento del sitio web se simplifica al máximo, ya que un solo cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todas las páginas HTML que enlazan ese archivo.

Aunque generalmente se emplea la etiqueta <link> para enlazar los archivos CSS externos, también se puede utilizar la etiqueta <style>.

La forma alternativa de incluir un archivo CSS externo se muestra a continuación:
Modificamos ejemplo anterior (**Ej01.06b-CSSExterno.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css" media="screen">
  @import 'css/Ej01.06b-CSSExterno.css';
  </style>
</head>
<body>
    <section>
      <p>Un párrafo de texto.</p>
    </section>
</body>
</html>
```

1.6.3 Incluir CSS en los elementos HTML

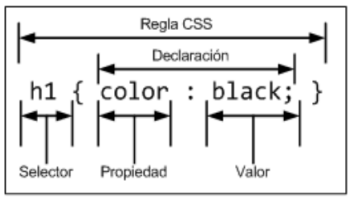
El último método para incluir estilos CSS en documentos HTML es el peor y el menos utilizado, ya que tiene los mismos problemas que la utilización de las etiquetas <font>.
Un ejemplo usando HTML4 (**Ej01.06c-CSSInterno.html)**:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
      <title>Ejemplo de estilos CSS en el propio documento</title>
   </head>

   <body>
      <p style="color: black; font-family: Verdana;">Un párrafo de texto.</p>
   </body>
</html>
```

Esta forma de incluir CSS directamente en los elementos HTML solamente se utiliza en determinadas situaciones en las que se debe incluir un estilo muy específico para un solo elemento concreto.

1.7 Glosario básico

CSS define una serie de términos que permiten describir cada una de las partes que componen los estilos CSS. El siguiente esquema muestra las partes que forman un estilo CSS muy básico:



Los diferentes términos se definen a continuación:

- **Regla:** cada uno de los estilos que componen una hoja de estilos CSS. Cada regla está compuesta de una parte de "selectores", un símbolo de "llave de apertura" ({), otra parte denominada "declaración" y por último, un símbolo de "llave de cierre" (}).
- **Selector:** indica el elemento o elementos HTML a los que se aplica la regla CSS.
- **Declaración:** especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- **Propiedad:** característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
- **Valor:** establece el nuevo valor de la característica modificada en el elemento.

Un archivo CSS puede contener un número ilimitado de reglas CSS, cada regla se puede aplicar a varios selectores diferentes y cada declaración puede incluir tantos pares propiedad/valor como se desee. El estándar CSS 2.1 define 115 propiedades, cada una con su propia lista de valores permitidos. Por su parte, los últimos borradores del estándar CSS 3 ya incluyen 239 propiedades.

1.8 Medios CSS

Una de las características más importantes de las hojas de estilos CSS es que permiten definir diferentes estilos para diferentes medios o dispositivos: pantallas, impresoras, móviles, proyectores, etc.

Además, CSS define algunas propiedades específicamente para determinados medios, como por ejemplo la paginación y los saltos de página para los medios impresos o el volumen y tipo de voz para los medios de audio. La siguiente tabla muestra el nombre que CSS utiliza para identificar cada medio y su descripción:

Medio	Descripción
all	Todos los medios definidos
braille	Dispositivos táctiles que emplean el sistema braille
embosed	Impresoras braille
handheld	Dispositivos de mano: móviles, PDA, etc.
print	Impresoras y navegadores en el modo *"Vista Previa para Imprimir"*
projection	Proyectores y dispositivos para presentaciones
screen	Pantallas de ordenador
speech	Sintetizadores para navegadores de voz utilizados por personas discapacitadas
tty	Dispositivos textuales limitados como teletipos y terminales de texto
tv	Televisores y dispositivos con resolución baja

os medios más utilizados actualmente son screen (para definir el aspecto de la página en pantalla) y print (para definir el aspecto de la página cuando se imprime), seguidos de handheld (que define el aspecto de la página cuando se visualiza mediante un dispositivo móvil).

Además, CSS clasifica a los medios en diferentes grupos según sus características. La siguiente tabla resume todos los grupos definidos en el estándar:

Medio	Continuo / Paginado	Visual / Auditivo / Táctil / Vocal	Mapa de bits / Caracteres	Interactivo / Estático
braille	continuo	táctil	caracteres	ambos
embossed	paginado	táctil	caracteres	estático
handheld	ambos	visual, auditivo, vocal	ambos	ambos
print	paginado	visual	mapa de bits	estático
projection	paginado	visual	mapa de bits	interactivo
screen	continuo	visual, auditivo	mapa de bits	ambos
speech	continuo	vocal	(no tiene sentido)	ambos
tty	continuo	visual	caracteres	ambos
tv	ambos	visual, auditivo	mapa de bits	ambos

La gran ventaja de CSS es que permite modificar los estilos de una página en función del medio en el que se visualiza. Existen cuatro formas diferentes de indicar el medio en el que se deben aplicar los estilos CSS.

1.8.1 Medios definidos con las reglas de tipo @media.

Las reglas @media son un tipo especial de regla CSS que permiten indicar de forma directa el medio o medios en los que se aplicarán los estilos incluidos en la regla. Para especificar el medio en el que se aplican los estilos, se incluye su nombre después de @media. Si los estilos se aplican a varios medios, se incluyen los nombres de todos los medios separados por comas.

Un ejemplo usando HTML5 (**Ej01.08a-Media.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
    @media print {
     body { font-size: 10pt; }
   }
   @media screen {
     body { font-size: 13px; }
   }
   @media screen, print {
     body { line-height: 1.2; }
   }

  </style>
</head>
<body>
    <section>
      <p>Párrafo de ejemplo.</p>
    </section>
</body>
</html>
```

El ejemplo anterior establece que el tamaño de letra de la página cuando se visualiza en una pantalla debe ser 13 píxel. Sin embargo, cuando se imprimen los contenidos de la página, su tamaño de letra debe ser de 10 puntos. Por último, tanto cuando la página se visualiza en una pantalla como cuando se imprimen sus contenidos, el interlineado del texto debe ser de 1.2 veces el tamaño de letra del texto.

1.8.2 Medios definidos con las reglas de tipo @import.

Cuando se utilizan reglas de tipo @import para enlazar archivos CSS externos, se puede especificar el medio en el que se aplican los estilos indicando el nombre del medio después de la URL del archivo CSS:

```
@import url("estilos_basicos.css") screen;
@import url("estilos_impresora.css") print;
```

Las reglas del ejemplo anterior establecen que cuando la página se visualiza por pantalla, se cargan los estilos definidos en el primer archivo CSS. Por otra parte, cuando la página se imprime, se tienen en cuenta los estilos que define el segundo archivo CSS.

Si los estilos del archivo CSS externo deben aplicarse en varios medios, se indican los nombres de todos los medios separados por comas. Si no se indica el medio en una regla de tipo @import, el navegador sobreentiende que el medio es all, es decir, que los estilos se aplican en todos los medios.

1.8.3 Medios definidos con la etiqueta

Si se utiliza la etiqueta <link> para enlazar los archivos CSS externos, se puede utilizar el atributo media para indicar el medio o medios en los que se aplican los estilos de cada archivo:

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css" />
<link rel="stylesheet" type="text/css" media="print, handheld"
      href="especial.css" />
```

En este ejemplo, el primer archivo CSS se tiene en cuenta cuando la página se visualiza en la pantalla (media="screen"). Los estilos indicados en el segundo archivo CSS, se aplican al imprimir la página (media="print") o al visualizarla en un dispositivo móvil (media="handheld"), como por ejemplo en un iPhone.

Si la etiqueta <link> no indica el medio CSS, se sobreentiende que los estilos se deben aplicar a todos los medios, por lo que es equivalente a indicar media="all".

1.8.4 Medios definidos mezclando varios métodos

CSS también permite mezclar los tres métodos anteriores para indicar los medios en los que se aplica cada archivo CSS externo:

```
<link rel="stylesheet" type="text/css"  media="screen" href="basico.css" />
@import url("estilos_seccion.css") screen;
@media print {  /* Estilos específicos para impresora */
}
```

Los estilos CSS que se aplican cuando se visualiza la página en una pantalla se obtienen mediante el recurso enlazado con la etiqueta <link> y mediante el archivo CSS externo incluido con la regla de tipo @import. Además, los estilos aplicados cuando se imprime la página se indican directamente en la página HTML mediante la regla de tipo @media.

1.9 Comentarios

CSS permite incluir comentarios entre sus reglas y estilos. Los comentarios son contenidos de texto que el diseñador incluye en el archivo CSS para su propia información y utilidad. Los navegadores ignoran por completo cualquier comentario de los archivos CSS, por lo que es común utilizarlos para estructurar de forma clara los archivos CSS complejos.

El comienzo de un comentario se indica mediante los caracteres /* y el final del comentario se indica mediante */, tal y como se muestra en el siguiente ejemplo:

```
/* Este es un comentario en CSS */
```

Los comentarios pueden ocupar tantas líneas como sea necesario, pero no se puede incluir un comentario dentro de otro comentario:

```
/* Este es un
   comentario CSS de varias
   lineas */
```

Aunque los navegadores ignoran los comentarios, su contenido se envía junto con el resto de estilos, por lo que no se debe incluir en ellos ninguna información sensible o confidencial.

La sintaxis de los comentarios CSS es muy diferente a la de los comentarios HTML, por lo que no deben confundirse:

```
<!-- Este es un comentario en HTML -->
<!-- Este es un
     comentario HTML de varias lineas -->
```

1.10 Sintaxis de la definición de cada propiedad CSS

A lo largo de los próximos capítulos, se incluyen las definiciones formales de la mayoría de propiedades de CSS. La definición formal se basa en la información recogida en el estándar oficial y se muestra en forma de tabla.

Una de las principales informaciones de cada definición es la lista de posibles valores que admite la propiedad. Para definir la lista de valores permitidos se sigue un formato que es necesario detallar.

Si el valor permitido se indica como una sucesión de palabras sin ningún carácter que las separe (paréntesis, comas, barras, etc.) el valor de la propiedad se debe indicar tal y como se muestra y con esas palabras en el mismo orden.

Si el valor permitido se indica como una sucesión de valores separados por una barra simple (carácter |) el valor de la propiedad debe tomar uno y sólo uno de los valores indicados.
Por ejemplo, la notación
`<porcentaje> | <medida> | inherit`
indica que la propiedad solamente puede tomar como valor la palabra reservada inherit o un porcentaje o una medida.

Si el valor permitido se indica como una sucesión de valores separados por una barra doble (símbolo ||) el valor de la propiedad puede tomar uno o más valores de los indicados y en cualquier orden.
Por ejemplo, la notación
`<color> || <estilo> || <medida>`
indica que la propiedad puede tomar como valor cualquier combinación de los valores indicados y en cualquier orden. Se podría establecer un color y un estilo, solamente una medida o una medida y un estilo. Además, el orden en el que se indican los valores es indiferente. Opcionalmente, se pueden utilizar paréntesis para agrupar diferentes valores.

Por último, en cada valor o agrupación de valores se puede indicar el tipo de valor: opcional, obligatorio, múltiple o restringido.

- El carácter * indica que el valor ocurre cero o más veces;
- el carácter + indica que el valor ocurre una o más veces;
- el carácter ? indica que el valor es opcional
- el carácter {número_1, número_2} indica que el valor ocurre al menos tantas veces como el valor indicado en número_1 y como máximo tantas veces como el valor indicado en número_2.

Por ejemplo, el valor
`[<family-name> , ]*`
indica que el valor de tipo <family_name> seguido por una coma se puede incluir cero o más veces.

El valor
`<url>? <color>`
significa que la URL es opcional y el color obligatorio y en el orden indicado.
Por último, el valor
`[<medida> | thick | thin] {1,4}`
indica que se pueden escribir entre 1 y 4 veces un valor que sea o una medida o la palabra thick o la palabra thin.

No obstante, la mejor forma de entender la notación formal para las propiedades de CSS es observar la definición de cada propiedad y volver a esta sección siempre que sea necesario.

2 SELECTORES

Para crear diseños web profesionales, es imprescindible conocer y dominar los selectores de CSS. Como se vio en el capítulo anterior, una regla de CSS está formada por una parte llamada "selector" y otra parte llamada "declaración".

La declaración indica "qué hay que hacer" y el selector indica "a quién hay que hacérselo". Por lo tanto, los selectores son imprescindibles para aplicar de forma correcta los estilos CSS en una página.

A un mismo elemento HTML se le pueden aplicar varias reglas CSS y cada regla CSS puede aplicarse a un número ilimitado de elementos. En otras palabras, una misma regla puede aplicarse sobre varios selectores y un mismo selector se puede utilizar en varias reglas.

El estándar de CSS 2.1 incluye una docena de tipos diferentes de selectores, que permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web. No obstante, la mayoría de páginas de los sitios web se pueden diseñar utilizando solamente los cinco selectores básicos.

2.1 Selectores básicos

2.1.1 Selector universal

Se utiliza para seleccionar todos los elementos de la página. El siguiente ejemplo elimina el margen y el relleno de todos los elementos HTML (por ahora no es importante fijarse en la parte de la declaración de la regla CSS):

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

El selector universal se indica mediante un asterisco (*). A pesar de su sencillez, no se utiliza habitualmente, ya que es difícil que un mismo estilo se pueda aplicar a todos los elementos de una página.

No obstante, sí que se suele combinar con otros selectores y además, forma parte de algunos hacks muy utilizados, como se verá más adelante.

2.1.2 Selector de tipo o etiqueta

Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. El siguiente ejemplo selecciona todos los párrafos de la página:

```
p {
  ...
}
```

Para utilizar este selector, solamente es necesario indicar el nombre de una etiqueta HTML (sin los caracteres < y >) correspondiente a los elementos que se quieren seleccionar.
El siguiente ejemplo aplica diferentes estilos a los titulares y a los párrafos de una página HTML:

```
h1 {
  color: red;
}
h2 {
  color: blue;
}
p {
  color: black;
}
```

Si se quiere aplicar los mismos estilos a dos etiquetas diferentes, se pueden encadenar los selectores. En el siguiente ejemplo, los títulos de sección h1, h2 y h3 comparten los mismos estilos:

```
h1 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}
h2 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}
h3 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}
```

En este caso, CSS permite agrupar todas las reglas individuales en una sola regla con un selector múltiple. Para ello, se incluyen todos los selectores separados por una coma (,) y el resultado es que la siguiente regla CSS es equivalente a las tres reglas anteriores:

```
h1, h2, h3 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}
```

En las hojas de estilo complejas, es habitual agrupar las propiedades comunes de varios elementos en una única regla CSS y posteriormente definir las propiedades específicas de esos mismos elementos. El siguiente ejemplo establece en primer lugar las propiedades comunes de los títulos de sección (color y tipo de letra) y a continuación, establece el tamaño de letra de cada uno de ellos:

Un ejemplo usando HTML5 (**Ej02.01a-SelectoresTipo.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>

  <style type="text/css">
    h1, h2, h3 {
      font-weight: bold;
      font-family: Arial, Helvetica, sans-serif;
    }
    h1 { color: blue; }
    h2 { color: #F00; }
    h3 { color: rgb(0,255,0); }
  </style>

</head>
<body>
    <section>
      <h1> Título Principal</h1>
      <h2> Apartado 1</h2>
      <h3> Sección A</h3>
      <p>Párrafo de ejemplo.</p>
    </section>
</body>
</html>
```

2.1.3 Selector descendente

Selecciona los elementos que se encuentran dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento.

El selector del siguiente ejemplo selecciona todos los elementos de la página que se encuentren dentro de un elemento <p>:

```
p span { color: red; }
```

Veamos ejemplo usando HTML5 (**Ej02.01b-SelectorDescendente.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
    p span { color: red; }
  </style>
</head>
<body>
    <section>
<p> Aquí tenemos parte del texto <span>en rojo</span>.
<br />
Incluso en un nivel inferior, como un
<a href="#"><span>enlace</span></a>
</p>
<br>
  Fuera del párrafo el <span> SPAN se mantiene negro</span>
    </section>
</body>
</html>
```

El selector p span selecciona tanto "en rojo" como "enlace".
El motivo es que en el selector descendente, un elemento no tiene que ser descendiente directo del otro. La única condición es que un elemento debe estar dentro de otro elemento, sin importar el nivel de profundidad en el que se encuentre.

Al resto de elementos de la página que no están dentro de un elemento <p>, y no se les aplica la regla CSS anterior.

Los selectores descendentes permiten aumentar la precisión del selector de tipo o etiqueta. Así, utilizando el selector descendente es posible aplicar diferentes estilos a los elementos del mismo tipo. El siguiente ejemplo amplía el anterior y muestra de color azul todo el texto de los contenidos dentro de un <h1>.
Otro ejemplo (**Ej02.01c-SelectorDescendentePH1.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      p span  { color: red;  }
      h1 span { color: blue; }
  </style>
</head>
<body>
    <section>
      <h1> <span>Titulo en Azul</span></h1>
      <p> Aquí tenemos parte del texto <span>en rojo</span>.
    </section>
</body>
</html>
```

Con las reglas CSS anteriores:

- Los elementos dentro de un elemento <p> se muestran de color rojo.
- Los elementos dentro de un elemento <h1> se muestran de color azul.
- El resto de elementos de la página, se muestran con el color por defecto aplicado por el navegador.

La sintaxis formal del selector descendente se muestra a continuación:

```
selector1 selector2 selector3 ... selectorN
```

Los selectores descendentes siempre están formados por dos o más selectores separados entre sí por espacios en blanco. El último selector indica el elemento sobre el que se aplican los estilos y todos los selectores anteriores indican el lugar en el que se debe encontrar ese elemento.

En el siguiente ejemplo, el selector descendente se compone de cuatro selectores:

```
p a span em { text-decoration: underline; }
```

Los estilos de la regla anterior se aplican a los elementos de tipo que se encuentren dentro de elementos de tipo , que a su vez se encuentren dentro de elementos de tipo <a> que se encuentren dentro de elementos de tipo <p>.

No debe confundirse el selector descendente con la combinación de selectores:

```
/* El estilo se aplica a todos los elementos "p", "a", "span" y "em" */
p, a, span, em { text-decoration: underline; }

/* Se aplica SOLO a los elementos "em" que se encuentran dentro de "p a span" */
p a span em { text-decoration: underline; }
```

Se puede restringir el alcance del selector descendente combinándolo con el selector universal. El siguiente ejemplo, muestra los dos enlaces de color rojo:

```
p a { color: red; }
...
<p><a href="#">Enlace</a></p>
<p><span><a href="#">Enlace</a></span></p>
```

Sin embargo, en el siguiente ejemplo solamente el segundo enlace se muestra de color rojo:

```
p * a { color: red; }
...
<p><a href="#">Enlace</a></p>
<p><span><a href="#">Enlace</a></span></p>
```

La razón es que el selector p * a se interpreta como todos los elementos de tipo <a> que se encuentren dentro de cualquier elemento que, a su vez, se encuentre dentro de un elemento de tipo <p>. Como el primer elemento <a> se encuentra directamente bajo un elemento <p>, no se cumple la condición del selector p * a. Ejemplo (**`Ej02.01d-SelectorDescendenteP.html`**):

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      p * a { color: red;  }
  </style>
</head>
<body>
    <section>
      <p><a href="#">Enlace</a></p>
      <p><span><a href="#">Enlace</a></span></p>
    </section>
</body>
</html>
```

2.1.4 Selector de clase

Si se considera el siguiente código HTML de ejemplo:

```
<body>
  <p>Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et est adipiscing accumsan...</p>
  <p>Class aptent taciti sociosqu ad litora...</p>
</body>
```

¿Cómo se pueden aplicar estilos CSS sólo al primer párrafo? El selector universal (*) no se puede utilizar porque selecciona todos los elementos de la página. El selector de tipo o etiqueta (p) tampoco se puede utilizar porque seleccionaría todos los párrafos. Por último, el selector descendente (body p) tampoco se puede utilizar porque todos los párrafos se encuentran en el mismo sitio.

Una de las soluciones más sencillas para aplicar estilos a un solo elemento de la página consiste en utilizar el atributo class de HTML sobre ese elemento para indicar directamente la regla CSS que se le debe aplicar:

```
<body>
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et est adipiscing accumsan...</p>
  <p>Class aptent taciti sociosqu ad litora...</p>
</body>
```

A continuación, se crea en el archivo CSS una nueva regla llamada destacado con todos los estilos que se van a aplicar al elemento. Para que el navegador no confunda este selector con los otros tipos de selectores, se prefija el valor del atributo class con un punto (.) tal y como muestra el siguiente ejemplo:

```
.destacado { color: red; }
```

Veamos un ejemplo con el CSS Interno (**`Ej02.01e-SelectorClase.html`**):
NOTA: Como ya hemos visto anteriormente, usamos el Lorem Ipsum → http://es.lipsum.com/

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      .destacado { color: red; }
  </style>
</head>
<body>
    <section>
      <p class="destacado">Lorem ipsum dolor sit amet...</p>
      <p>Nunc sed lacus et est adipiscing accumsan...</p>
      <p>Class aptent taciti sociosqu ad litora...</p>
    </section>
</body>
</html>
```

El selector .destacado se interpreta como "cualquier elemento de la página cuyo atributo class sea igual a destacado", por lo que solamente el primer párrafo cumple esa condición. Este tipo de selectores se llaman selectores de clase y son los más usados junto con los selectores de ID que se verán luego. La principal característica de este selector es que en una misma página HTML varios elementos diferentes pueden utilizar el mismo valor en el atributo class:

```
<body>
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a>
accumsan...</p>
  <p>Class aptent taciti <em class="destacado">sociosqu ad</em> litora...</p>
</body>
```

Los selectores de clase son imprescindibles para diseñar páginas web complejas, ya que permiten disponer de una precisión total al seleccionar los elementos. Además, estos selectores permiten reutilizar los mismos estilos para varios elementos diferentes.

A continuación se muestra otro ejemplo de selectores de clase:

```
.aviso {
  padding: 0.5em;
  border: 1px solid #98be10;
  background: #f6feda;
}
 .error {
  color: #930;
  font-weight: bold;
}
...

<span class="error">...</span>
<div class="aviso">...</div>
```

El elemento tiene un atributo class="error", por lo que se le aplican las reglas CSS indicadas por el selector .error. Por su parte, el elemento tiene un atributo class="aviso", por lo que su estilo es el que definen las reglas CSS del selector .aviso.

En ocasiones, es necesario restringir el alcance del selector de clase. Si se considera de nuevo el ejemplo anterior:

```
<body>
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a>
accumsan...</p>
  <p>Class aptent taciti <em class="destacado">sociosqu ad</em> litora...</p>
</body>
```

¿Cómo es posible aplicar estilos solamente al párrafo cuyo atributo class sea igual a destacado? Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico:

```
p.destacado { color: red }
```

El selector p.destacado se interpreta como "aquellos elementos de tipo <p> que dispongan de un atributo class con valor destacado". De la misma forma, el selector a.destacado solamente selecciona los enlaces cuyo atributo class sea igual a destacado.

De lo anterior se deduce que el atributo .destacado es equivalente a *.destacado, por lo que todos los diseñadores obvian el símbolo * al escribir un selector de clase normal.
Veamos el ejemplo anterior (**Ej02.01f-SelectorClase.html**):

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      p.destacado { color: red }
  </style>
</head>
<body>
    <section>
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a> accumsan...</p>
  <p>Class aptent taciti <em class="destacado">sociosqu ad</em> litora...</p>
    </section>
</body>
</html>
```

No debe confundirse el selector de clase con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo class="aviso" */
   p.aviso { ... }

/* Todos los elementos con atributo class="aviso" que estén dentro
   de cualquier elemento de tipo "p" */
   p .aviso { ... }

/* Todos los elementos "p" de la página y todos los elementos con
   atributo class="aviso" de la página */
   p, .aviso { ... }
```

Por último, es posible aplicar los estilos de varias clases CSS sobre un mismo elemento. La sintaxis es similar, pero los diferentes valores del atributo class se separan con espacios en blanco. En el siguiente ejemplo:

```
<p class="especial destacado error">Párrafo de texto...</p>
```

Al párrafo anterior se le aplican los estilos definidos en las reglas .especial, .destacado y .error, por lo que en el siguiente ejemplo, el texto del párrafo se vería de color rojo, en negrita y con un tamaño de letra de 15 píxel:

```
.error { color: red; }
.destacado { font-size: 15px; }
.especial  { font-weight: bold; }
```

Veamos el ejemplo anterior (**Ej02.01f-SelectorClase.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      .error { color: red; }
      .destacado { font-size: 15px; }
      .especial  { font-weight: bold; }
  </style>
</head>
<body>
    <section>
      <p class="especial destacado error">Párrafo de texto...</p>
    </section>
</body>
</html>
```

Si un elemento dispone de un atributo class con más de un valor, es posible utilizar un selector más avanzado:

```
.error { color: red; }
.error.destacado { color: blue; }
.destacado { font-size: 15px; }
.especial  { font-weight: bold; }

...
<p class="especial destacado error">Párrafo de texto...</p>
```

En el ejemplo anterior, el color de la letra del texto es azul y no rojo. El motivo es que se ha utilizado un selector de clase múltiple .error.destacado, que se interpreta como "aquellos elementos de la página que dispongan de un atributo class con al menos los valores error y destacado".

2.1.5 Selectores de ID

En ocasiones, es necesario aplicar estilos CSS a un único elemento de la página. Aunque puede utilizarse un selector de clase para aplicar estilos a un único elemento, existe otro selector más eficiente en este caso. El selector de ID permite seleccionar un elemento de la página a través del valor de su atributo id. Este tipo de selectores sólo seleccionan un elemento de la página porque el valor del atributo id no se puede repetir en dos elementos diferentes de una misma página.

La sintaxis de los selectores de ID es muy parecida a la de los selectores de clase, salvo que se utiliza el símbolo de la almohadilla (#) en vez del punto (.) como prefijo del nombre de la regla CSS. Veamos un ejemplo (**Ej02.01g-SelectorID.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      #destacado { color: red; }
  </style>
</head>
<body>
    <section>
      <p>Primer párrafo</p>
      <p id="destacado">Segundo párrafo</p>
      <p>Tercer párrafo</p>
    </section>
</body>
</html>
```

En el ejemplo anterior, el selector #destacado solamente selecciona el segundo párrafo (cuyo atributo id es igual a destacado).

La principal diferencia entre este tipo de selector y el selector de clase tiene que ver con HTML y no con CSS. Como se sabe, en una misma página, el valor del atributo id debe ser único, de forma que dos elementos diferentes no pueden tener el mismo valor de id. Sin embargo, el atributo class no es obligatorio que sea único, de forma que muchos elementos HTML diferentes pueden compartir el mismo valor para su atributo class.

De esta forma, la recomendación general es la de utilizar el selector de ID cuando se quiere aplicar un estilo a un solo elemento específico de la página y utilizar el selector de clase cuando se quiere aplicar un estilo a varios elementos diferentes de la página HTML.

Al igual que los selectores de clase, en este caso también se puede restringir el alcance del selector mediante la combinación con otros selectores. El siguiente ejemplo aplica la regla CSS solamente al elemento de tipo <p> que tenga un atributo id igual al indicado:

```
p#aviso { color: blue; }
```

A primera vista, restringir el alcance de un selector de ID puede parecer absurdo. En realidad, un selector de tipo p#aviso sólo tiene sentido cuando el archivo CSS se aplica sobre muchas páginas HTML diferentes.

En este caso, algunas páginas pueden disponer de elementos con un atributo id igual a aviso y que no sean párrafos, por lo que la regla anterior no se aplica sobre esos elementos.
No debe confundirse el selector de ID con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo id="aviso" */
p#aviso { ... }
/* Todos los elementos con atributo id="aviso" dentro de cualquier elemento de tipo "p" */
p #aviso { ... }
/* Todos los elementos "p" de la página y todos los elementos con atributo id="aviso" */
p, #aviso { ... }
```

2.1.6 Combinación de selectores básicos

CSS permite la combinación de uno o más tipos de selectores para restringir el alcance de las reglas CSS. A continuación se muestran algunos ejemplos habituales de combinación de selectores.

```
.aviso .especial { ... }
```

El anterior selector solamente selecciona aquellos elementos con un class="especial" que se encuentren dentro de cualquier elemento con un class="aviso".

Si se modifica el anterior selector:

```
div.aviso span.especial { ... }
```

Ahora, el selector solamente selecciona aquellos elementos de tipo con un atributo class="especial" que estén dentro de cualquier elemento de tipo que tenga un atributo class="aviso".

La combinación de selectores puede llegar a ser todo lo compleja que sea necesario:

```
ul#menuPrincipal li.destacado a#inicio { ... }
```

El anterior selector hace referencia al enlace con un atributo id igual a inicio que se encuentra dentro de un elemento de tipo <li> con un atributo class igual a destacado, que forma parte de una lista <ul> con un atributo id igual a menuPrincipal.

Ejercicio 01

A partir del código HTML y CSS que se muestra, añadir los selectores CSS que faltan para aplicar los estilos deseados. Cada regla CSS incluye un comentario en el que se explica los elementos a los que debe aplicarse:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>Ejercicio de selectores</title>
<style type="text/css">
/* Todos los elementos de la pagina */
{ font: 1em/1.3 Arial, Helvetica, sans-serif; }

/* Todos los parrafos de la pagina */
{ color: #555; }

/* Todos los párrafos contenidos en #primero */
{ color: #336699; }

/* Todos los enlaces la pagina */
{ color: #CC3300; }

/* Los elementos "em" contenidos en #primero */
{ background: #FFFFCC; padding: .1em; }

/* Todos los elementos "em" de clase "especial" en toda la pagina */
{ background: #FFCC99; border: 1px solid #FF9900; padding: .1em; }

/* Elementos "span" contenidos en .normal */
{ font-weight: bold; }

</style>
</head>

<body>

<div id="primero">
<p>Lorem ipsum dolor sit amet, <a href="#">consectetuer adipiscing elit</a>. Praesent
blandit nibh at felis. Sed nec diam in dolor vestibulum aliquet. Duis ullamcorper, nisi non
facilisis molestie, <em>lorem sem aliquam nulla</em>, id lacinia velit mi vestibulum
enim.</p>

</div>

<div class="normal">
<p>Phasellus eu velit sed lorem sodales egestas. Ut feugiat. <span><a href="#">Donec
porttitor</a>, magna eu varius luctus,</span> metus massa tristique massa, in imperdiet est
velit vel magna. Phasellus erat. Duis risus. <a href="#">Maecenas dictum</a>, nibh vitae
pellentesque auctor, tellus velit consectetuer tellus, tempor pretium felis tellus at
metus.</p>

<p>Cum sociis natoque <em class="especial">penatibus et magnis</em> dis parturient montes,
nascetur ridiculus mus. Proin aliquam convallis ante. Pellentesque habitant morbi tristique
senectus et netus et malesuada fames ac turpis egestas. Nunc aliquet. Sed eu metus. Duis
justo.</p>

<p>Donec facilisis blandit velit. Vestibulum nisi. Proin volutpat, <em
class="especial">enim id iaculis congue</em>, orci justo ultrices tortor, <a href="#">quis
lacinia eros libero in eros</a>. Sed malesuada dui vel quam. Integer at eros.</p>
</div>

</body>
</html>
```

2.2 Selectores avanzados

Utilizando solamente los selectores básicos de la sección anterior, es posible diseñar prácticamente cualquier página web. No obstante, CSS define otros selectores más avanzados que permiten simplificar las hojas de estilos.

Desafortunadamente, el navegador Internet Explorer 6 y sus versiones anteriores no soportaban este tipo de selectores avanzados, por lo que su uso no era común hasta hace poco tiempo. Si quieres consultar el soporte de los selectores en los distintos navegadores, puedes utilizar las siguientes referencias:

- Lista completa de los selectores que soporta cada versión de cada navegador http://dev.l-c-n.com/CSS3-selectors/browser-support.php
- Test online para comprobar el soporte del navegador empleado http://www.css3.info/selectors-test/

2.2.1 Selector de hijos

Se trata de un selector similar al selector descendente, pero muy diferente en su funcionamiento. Se utiliza para seleccionar un elemento que es hijo directo de otro elemento y se indica mediante el "signo de mayor que" (>):

```
p > span { color: blue; }

<p><span>Texto1</span></p>
<p><a href="#"><span>Texto2</span></a></p>
```

En el ejemplo anterior, el selector p > span se interpreta como "cualquier elemento que sea hijo directo de un elemento <p>", por lo que el primer elemento cumple la condición del selector. Sin embargo, el segundo elemento no la cumple porque es descendiente pero no es hijo directo de un elemento <p>.

El siguiente ejemplo muestra las diferencias entre el selector descendente y el selector de hijos: (**Ej02.02a-SelectorHijo.html**):

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      p a { color: red; }
      p > a { color: blue; }
  </style>
</head>
<body>
    <section>
      <p><a href="#">Enlace1</a></p>
      <p><span><a href="#">Enlace2</a></span></p>
    </section>
</body>
</html>
```

Obsérvese como en el primer caso, al ser un selector descendente, se aplica al segundo enlace (aunque esté span por medio), mientras que p>a nos indica un enlace hijo directo de p, que es el primer caso.

2.2.2 Selector adyacente

El selector adyacente se emplea para seleccionar elementos que en el código HTML de la página se encuentran justo a continuación de otros elementos. Su sintaxis emplea el signo + para separar los dos elementos:

```
elemento1 + elemento2 { ... }
```

Si se considera el siguiente código HTML:

```
<body>
<h1>Titulo1</h1>
<h2>Subtítulo</h2>
...
<h2>Otro subtítulo</h2>
...
</body>
```

La página anterior dispone de dos elementos <h2>, pero sólo uno de ellos se encuentra inmediatamente después del elemento <h1>. Si se quiere aplicar diferentes colores en función de esta circunstancia, el selector adyacente es el más adecuado:

```
h2 { color: green; }
h1 + h2 { color: red }
```

Las reglas CSS anteriores hacen que todos los <h2> de la página se vean de color verde, salvo aquellos <h2> que se encuentran inmediatamente después de cualquier elemento <h1> y que se muestran de color rojo.

Técnicamente, los elementos que forman el selector adyacente deben cumplir las dos siguientes condiciones:

- elemento1 y elemento2 deben ser elementos hermanos, por lo que su elemento padre debe ser el mismo.
- elemento2 debe aparecer inmediatamente después de elemento1 en el código HTML de la página.

El siguiente ejemplo es muy útil para los textos que se muestran como libros:

```
p + p { text-indent: 1.5em; }
```

En muchos libros, suele ser habitual que la primera línea de todos los párrafos esté indentada, salvo la primera línea del primer párrafo. Con el selector p + p, se seleccionan todos los párrafos de la página que estén precedidos por otro párrafo, por lo que no se aplica al primer párrafo de la página.

(**Ej02.02b-SelectorAdyacente.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      h2 { color: green; }
      h1 + h2 { color: red }
  </style>
</head>
<body>
    <section>
      <h1>Titulo Principal</h1>
      <h2>Subtítulo1</h2>
      <p></p><br /><p></p><br />
      <h2>Subtítulo2</h2>
    </section>
</body>
</html>
```

2.2.3 Selector de atributos.

El último tipo de selectores avanzados lo forman los selectores de atributos, que permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

Los cuatro tipos de selectores de atributos son:

- **[nombre_atributo]**, selecciona los elementos que tienen establecido el atributo llamado nombre_atributo, independientemente de su valor.
- **[nombre_atributo=valor]**, selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con un valor igual a valor.
- **[nombre_atributo~=valor]**, selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y al menos uno de los valores del atributo es valor.
- **[nombre_atributo|=valor]**, selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y cuyo valor es una serie de palabras separadas con guiones, pero que comienza con valor. Este tipo de selector sólo es útil para los atributos de tipo lang que indican el idioma del contenido del elemento.

A continuación se muestran algunos ejemplos de estos tipos de selectores:
(**Ej02.02c-SelectorAtributo.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
/* Se muestran de color verde todos los enlaces que tengan
   un atributo "id", independientemente de su valor */
a[id] { color: green; }

/* Se muestran de color amarillo todos los enlaces que tengan
   un atributo "class" con el valor "externo" */
a[class="externo"] { color: yellow; }

/* Se muestran de color violeta todos los enlaces que apunten
   al sitio "http://www.ejemplo.com" */
a[href="http://www.ejemplo.com"] { color: purple  ; }

/* Se muestran de color naranja todos los enlaces que tengan un atributo "class"
 en el que al menos uno de sus valores sea "externo" */
a[class~="abajo"] { color: orange; }

/* Selecciona todos los elementos de la página cuyo atributo
   "lang" sea igual a "en", es decir, todos los elementos en inglés */
*[lang=en] { color: brown; }

/* Selecciona todos los elementos de la página cuyo atributo
   "lang" empiece por "es", es decir, "es", "es-ES", "es-AR", etc. */
*[lang|="es"] { color : red }
  </style>
</head>
<body>
    <section>
      <a id="interno" href="#">Enlace Interno</a> <br />
      <a class="externo" href="#">Enlace Externo</a> <br />
      <a class="arriba abajo" href="#">Enlace Arriba/Abajo</a> <br />
      <a href="http://www.ejemplo.com">Enlace Ejemplo</a> <br />
      <a lang="en" href="#">Enlace Inglés</a> <br />
      <a lang="es-ar" href="#">Enlace Español</a> <br />
    </section>
</body>
</html>
```

2.3 Agrupación de reglas

Cuando se crean archivos CSS complejos con decenas o cientos de reglas, es habitual que los estilos que se aplican a un mismo selector se definan en diferentes reglas:

```
h1 { color: red; }
...
h1 { font-size: 2em; }
...
h1 { font-family: Verdana; }
```

Las tres reglas anteriores establecen el valor de tres propiedades diferentes de los elementos <h1>. Antes de que el navegador muestre la página, procesa todas las reglas CSS de la página para tener en cuenta todos los estilos definidos para cada elemento.

Cuando el selector de dos o más reglas CSS es idéntico, se pueden agrupar las declaraciones de las reglas para hacer las hojas de estilos más eficientes:

```
h1 {
  color: red;
  font-size: 2em;
  font-family: Verdana;
}
```

El ejemplo anterior tiene el mismo efecto que las tres reglas anteriores, pero es más eficiente y es más fácil de modificar y mantener por parte de los diseñadores. Como CSS ignora los espacios en blanco y las nuevas líneas, también se pueden agrupar las reglas de la siguiente forma:

```
h1 { color: red; font-size: 2em; font-family: Verdana; }
```

Si se quiere reducir al máximo el tamaño del archivo CSS para mejorar ligeramente el tiempo de carga de la página web, también es posible indicar la regla anterior de la siguiente forma:

```
h1 {color:red;font-size:2em;font-family:Verdana;}
```

De todos modos, el mas recomendado es la agrupación multilínea, como en el siguiente ejemplo: **Ej02.03-AgruparReglas.html)**:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
   h1 {
      color: red;
      font-size: 2em;
      font-family: Verdana;
   }
  </style>
</head>
<body>
    <section>
      <h1>Titulo Principal</h1>
    </section>
</body>
</html>
```

2.4 Herencia

Una de las características principales de CSS es la herencia de los estilos definidos para los elementos. Cuando se establece el valor de una propiedad CSS en un elemento, sus elementos descendientes heredan de forma automática el valor de esa propiedad. Si se considera el siguiente ejemplo: **Ej02.04a-Herencia.html)**:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      body { color: blue; }
  </style>
</head>
<body>
    <section>
      <h1>Titulo Principal</h1>
      <p>Un párrafo de texto no muy largo.</p>
    </section>
</body>
</html>
```

En el ejemplo anterior, el selector body solamente establece el color de la letra para el elemento <body>. No obstante, la propiedad color es una de las que se heredan de forma automática, por lo que todos los elementos descendientes de <body> muestran ese mismo color de letra. Por tanto, establecer el color de la letra en el elemento <body> de la página implica cambiar el color de letra de todos los elementos de la página.

Aunque la herencia de estilos se aplica automáticamente, se puede anular su efecto estableciendo de forma explícita otro valor para la propiedad que se hereda, como se muestra en el siguiente ejemplo:**Ej02.04b-HerenciaModificada.html)**:

```
<!DOCTYPE html> <html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Herencia</title>
   <style>
      body {color: blue;     font-family: Arial;}
      p { font-family: cursive;}     /* Los párrafos los pongo en Comic Sans*/
      span {color: red;}
   </style>
</head>
<body>
<section>
      <h1>Titulo de Sección</h1>
      <p> Esto es un párrafo con <span> TEXTO ESPECIAL</span></p>
</section>
</body> </html>
```

En el ejemplo anterior, se establece en primer lugar el color y tipo de letra del elemento <body>, por lo que todos los elementos de la página se mostrarían con ese mismo color y tipo de letra. No obstante, las otras reglas CSS modifican alguno de los estilos heredados. De esta forma, los elementos <h1> de la página se muestran con el tipo de letra Verdana establecido por el selector h1 y se muestran de color negro que es el valor heredado del elemento <body>. Igualmente, los elementos <p> de la página se muestran del color rojo establecido por el selector p y con un tipo de letra Arial heredado del elemento <body>.

La mayoría de propiedades CSS aplican la herencia de estilos de forma automática. Además, para aquellas propiedades que no se heredan automáticamente, CSS incluye un mecanismo para forzar a que se hereden sus valores, tal y como se verá más adelante. Por último, aunque la herencia automática de estilos puede parecer complicada, simplifica en gran medida la creación de hojas de estilos complejas. Como se ha visto en los ejemplos anteriores, si se quiere establecer por ejemplo la tipografía base de la página, simplemente se debe establecer en el elemento <body> de la página y el resto de elementos la heredarán de forma automática.

2.5 Colisiones de estilos

En las hojas de estilos complejas, es habitual que varias reglas CSS se apliquen a un mismo elemento HTML. El problema de estas reglas múltiples es que se pueden dar colisiones como la del siguiente ejemplo:

```
p { color: red; }
p { color: blue; }
...
<p>...</p>
```

¿De qué color se muestra el párrafo anterior? CSS tiene un mecanismo de resolución de colisiones muy complejo y que tiene en cuenta el tipo de hoja de estilo que se trate (de navegador, de usuario o de diseñador), la importancia de cada regla y lo específico que sea el selector.

El método seguido por CSS para resolver las colisiones de estilos se muestra a continuación:

- Determinar todas las declaraciones que se aplican al elemento para el medio CSS seleccionado.
- Ordenar las declaraciones según su origen (tiene prioridad el CSS dentro del HTML, luego el CSS dentro del <style> y por último el CSS Externo) y su prioridad (palabra clave !important).
- Ordenar las declaraciones según lo específico que sea el selector. Cuanto más genérico es un selector, menos importancia tienen sus declaraciones.
- Si después de aplicar las normas anteriores existen dos o más reglas con la misma prioridad, se aplica la que se indicó en último lugar.

Hasta que no se expliquen más adelante los conceptos de tipo de hoja de estilo y la prioridad, el mecanismo simplificado que se puede aplicar es el siguiente:

- Cuanto más específico sea un selector, más importancia tiene su regla asociada.
- A igual especificidad, se considera la última regla indicada.

Como en el ejemplo anterior los dos selectores son idénticos, las dos reglas tienen la misma prioridad y prevalece la que se indicó en último lugar, por lo que el párrafo se muestra de color azul. En el siguiente ejemplo, la regla CSS que prevalece se decide por lo específico que es cada selector:**Ej02.05-ColisionesCSS.html)**:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
      p { color: red; }
      p#especial { color: green; }
      * { color: blue; }
  </style>
</head>
<body>
    <section>
      <h1>Titulo Principal</h1>
      <p id="especial">...</p>
    </section>
</body>
</html>
```

Al elemento <p> se le aplican las tres declaraciones. Como su origen y su importancia es la misma, decide la especificidad del selector. El selector * es el menos específico, ya que se refiere a "todos los elementos de la página". El selector p es poco específico porque se refiere a "todos los párrafos de la página". Por último, el selector p#especial sólo hace referencia a "el párrafo de la página cuyo atributo id sea igual a especial". Como el selector p#especial es el más específico, su declaración es la que se tiene en cuenta y por tanto el párrafo se muestra de color verde.

Ejercicio2

A partir del código HTML proporcionado, añadir las reglas CSS externas necesarias para que la página resultante tenga el mismo aspecto que el de la siguiente imagen:

Lorem ipsum dolor sit amet

Nulla pretium. Sed tempus nunc vitae neque. **Suspendisse gravida**, metus a scelerisque sollicitudin, lacus velit ultricies nisl, nonummy tempus neque diam quis felis. Etiam sagittis tortor sed arcu sagittis tristique.

Aliquam tincidunt, sem eget volutpat porta

Vivamus velit dui, placerat vel, feugiat in, ornare et, urna. Aenean turpis metus, *aliquam non*, tristique in, pretium varius, sapien. Proin vitae nisi. Suspendisse porttitor purus ac elit. Suspendisse eleifend odio at dui. In in elit sed metus pretium elementum.

Título de la tabla

	Título columna 1	Título columna 2
Título fila 1	Donec purus ipsum	Curabitur *blandit*
Título fila 2	Donec purus ipsum	Curabitur blandit
	Título columna 1	Título columna 2

Donec purus ipsum, posuere id, venenatis at, placerat ac, lorem. Curabitur blandit, eros sed gravida aliquet, risus justo porta lorem, ut mollis lectus tortor in orci. Pellentesque nec augue.

Fusce nec felis eu diam pretium adipiscing. Nunc elit elit, vehicula vulputate, venenatis in, posuere id, lorem. Etiam sagittis, tellus in ultrices accumsan, diam nisi feugiat ante, eu congue magna mi non nisl.

Vivamus ultrices aliquet augue. Donec arcu pede, pretium vitae, rutrum aliquet, tincidunt blandit, pede. Aliquam in nisi. Suspendisse volutpat. Nulla facilisi. Ut ullamcorper nisi quis mi.

A continuación se muestra el código HTML de la página sin estilos:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
/* Los estilos ...*/
  </style>
</head>
<body>
<h1 id="titulo">Lorem ipsum dolor sit amet</h1>

<p>Nulla pretium. Sed tempus nunc vitae neque. <strong>Suspendisse gravida</strong>, metus
a scelerisque sollicitudin, lacus velit
ultricies nisl, nonummy tempus neque diam quis felis. <span class="destacado">Etiam
sagittis tortor</span> sed arcu sagittis tristique.</p>

<h2 id="subtitulo">Aliquam tincidunt, sem eget volutpat porta</h2>

<p>Vivamus velit dui, placerat vel, feugiat in, ornare et, urna.  <a href="#">Aenean turpis
metus, <em>aliquam non</em>, tristique in</a>, pretium varius, sapien. Proin vitae nisi.
Suspendisse <span class="especial">porttitor purus ac elit</span>. Suspendisse eleifend
odio at dui. In in elit sed metus pretium elementum.</p>
```

```
<table summary="Descripción de la tabla y su contenido">
<caption>Título de la tabla</caption>
<thead>
  <tr>
    <th scope="col"></th>
    <th scope="col" class="especial">Título columna 1</th>
    <th scope="col" class="especial">Título columna 2</th>
  </tr>
</thead>

<tfoot>
  <tr>
    <th scope="col"></th>
    <th scope="col">Título columna 1</th>
    <th scope="col">Título columna 2</th>
  </tr>
</tfoot>

<tbody>
  <tr>
    <th scope="row" class="especial">Título fila 1</th>
    <td>Donec purus ipsum</td>
    <td>Curabitur <em>blandit</em></td>
  </tr>
  <tr>
    <th scope="row">Título fila 2</th>
    <td>Donec <strong>purus ipsum</strong></td>
    <td>Curabitur blandit</td>
  </tr>
</tbody>
</table>

<div id="adicional">
<p>Donec purus ipsum, posuere id, venenatis at, <span>placerat ac, lorem</span>. Curabitur
blandit, eros sed gravida aliquet, risus justo
porta lorem, ut mollis lectus tortor in orci. Pellentesque nec augue.</p>

<p>Fusce nec felis eu diam pretium adipiscing. <span id="especial">Nunc elit elit, vehicula
vulputate</span>, venenatis in,
posuere id, lorem. Etiam sagittis, tellus in ultrices accumsan, diam nisi feugiat ante, eu
congue magna mi non nisl.</p>

<p>Vivamus ultrices aliquet augue. <a href="#">Donec arcu pede, pretium vitae</a>, rutrum
aliquet, tincidunt blandit, pede.
Aliquam in nisi. Suspendisse volutpat. Nulla facilisi. Ut ullamcorper nisi quis mi.</p>
</div>

</body>
</html>
```

Aunque la propiedad que modifica el color del texto se explica detalladamente en los próximos capítulos, en este ejercicio solamente es preciso conocer que la propiedad se llama color y que como valor se puede indicar directamente el nombre del color.

Los nombres de los colores también están estandarizados y se corresponden con el nombre en inglés de cada color. En este ejercicio, se deben utilizar los colores: teal, red, blue, orange, purple, olive, fuchsia y green.

3 UNIDADES DE MEDIDA Y COLORES

Muchas de las propiedades de CSS que se ven en los próximos capítulos permiten indicar medidas y colores en sus valores. Además, CSS es tan flexible que permite indicar las medidas y colores de muchas formas diferentes. Por este motivo, se presentan a continuación todas las alternativas disponibles en CSS para indicar las medidas y los colores.

3.1 Unidades de medida

Las medidas en CSS se emplean, entre otras, para definir la altura, anchura y márgenes de los elementos y para establecer el tamaño de letra del texto. Todas las medidas se indican como un valor numérico entero o decimal seguido de una unidad de medida (sin ningún espacio en blanco entre el número y la unidad de medida).

CSS divide las unidades de medida en dos grupos: absolutas y relativas. Las medidas relativas definen su valor en relación con otra medida, por lo que para obtener su valor real, se debe realizar alguna operación con el valor indicado. Las unidades absolutas establecen de forma completa el valor de una medida, por lo que su valor real es directamente el valor indicado.

Si el valor es 0, la unidad de medida es opcional. Si el valor es distinto a 0 y no se indica ninguna unidad, la medida se ignora completamente, lo que suele ser uno de los errores más habituales de los diseñadores que empiezan con CSS. Algunas propiedades permiten indicar medidas negativas, aunque habitualmente sus valores son positivos. Si el valor decimal de una medida es inferior a 1, se puede omitir el 0 de la izquierda (0.5em es equivalente a .5em).

3.1.1 Unidades absolutas

Una medida indicada mediante unidades absolutas está completamente definida, ya que su valor no depende de otro valor de referencia. A continuación se muestra la lista completa de unidades absolutas definidas por CSS y su significado:

- **in**, pulgadas ("inches", en inglés). Una pulgada equivale a 2.54 centímetros.
- **cm**, centímetros.
- **mm**, milímetros.
- **pt**, puntos. Un punto equivale a 1 pulgada/72, es decir, unos 0.35 milímetros.
- **pc**, picas. Una pica equivale a 12 puntos, es decir, unos 4.23 milímetros.

Un Ejemplo: **Ej03.01a-UnidadesAbsolutas.html**:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
    body { margin: 0.5in; }  /* Cuerpo con margen de media pulgada */
    h1 { line-height: 2cm; }  /* h1 con un interlineado de 2 cms */
    p { word-spacing: 4mm; }  /* Espacido entre palabras 4 milímetros entre si */
    a { font-size: 18pt }     /* Enlaces con tamaño de letra de 18 puntos */
    span { font-size: 1pc }   /* Los <span> con tamaño de letra de 1 pica */
  </style>
</head>
<body>
    <section>
      <h1>Titulo Principal <br />
      Blog de la Clase </h1>
      <p>En este párrafo hay un <a href="#">Enlace</a> vacío.</p>
                <p>Aquí hay <span> parte del texto </span> diferente.</p>
                Texto fuera de P.
    </section>
</body>
</html>
```

La principal ventaja de las unidades absolutas es que su valor es directamente el valor que se debe utilizar, sin necesidad de realizar cálculos intermedios. Su principal desventaja es que son muy poco flexibles y no se adaptan fácilmente a los diferentes medios.

De todas las unidades absolutas, la única que suele utilizarse es el punto (pt). Se trata de la unidad de medida preferida para establecer el tamaño del texto en los documentos que se van a imprimir, es decir, para el medio print de CSS, tal y como se verá más adelante.

3.1.2 Unidades relativas

La unidades relativas, a diferencia de las absolutas, no están completamente definidas, ya que su valor siempre está referenciado respecto a otro valor. A pesar de su aparente dificultad, son las más utilizadas en el diseño web por la flexibilidad con la que se adaptan a los diferentes medios.

A continuación se muestran las tres unidades de medida relativas definidas por CSS y la referencia que toma cada una para determinar su valor real:

- **em**, (no confundir con la etiqueta) relativa respecto del tamaño de letra del elemento.
- **ex**, relativa respecto de la altura de la letra x ("equis minúscula") del tipo y tamaño de letra del elemento.
- **px, (píxel)** relativa respecto de la resolución de la pantalla del dispositivo en el que se visualiza la página HTML.

Las unidades em y ex no han sido creadas por CSS, sino que llevan décadas utilizándose en el campo de la tipografía. Aunque no es una definición exacta, la unidad 1em equivale a la anchura de la letra M ("eme mayúscula") del tipo y tamaño de letra del elemento.

La unidad em hace referencia al tamaño en puntos de la letra que se está utilizando. Si se utiliza una tipografía de 12 puntos, 1em equivale a 12 puntos. El valor de 1ex se puede aproximar por 0.5 em. Si se considera el siguiente ejemplo:

```
p { margin: 1em; }
```

La regla CSS anterior indica que los párrafos deben mostrar un margen de anchura igual a 1em. Como se trata de una unidad de medida relativa, es necesario realizar un cálculo matemático para determinar la anchura real de ese margen.

La unidad de medida em siempre hace referencia al tamaño de letra del elemento. Por otra parte, todos los navegadores muestran por defecto el texto de los párrafos con un tamaño de letra de 16 píxel. Por tanto, en este caso el margen de 1em equivale a un margen de anchura 16px. A continuación se modifica el ejemplo anterior para cambiar el tamaño de letra de los párrafos:

```
p { font-size: 32px; margin: 1em; }
```

El valor del margen sigue siendo el mismo en unidades relativas (1em) pero su valor real ha variado porque el tamaño de letra de los párrafos ha variado. En este caso, el margen tendrá una anchura de 32px, ya que 1em siempre equivale al tamaño de letra del elemento. Si se quiere reducir la anchura del margen a 16px pero manteniendo el tamaño de letra de los párrafos en 32px, se debe utilizar la siguiente regla CSS:

```
p { font-size: 32px; margin: 0.5em; }
```

El valor 0.5em se interpreta como "la mitad del tamaño de letra del elemento", ya que se debe multiplicar por 0.5 su tamaño de letra (32px x 0.5 = 16px). De la misma forma, si se quiere mostrar un margen de 8px de anchura, se debería utilizar el valor 0.25em, ya que 32px x 0.25 = 8px.

La gran ventaja de las unidades relativas es que siempre mantienen las proporciones del diseño de la página. Establecer el margen de un elemento con el valor 1em equivale a indicar que "el margen del elemento debe ser del mismo tamaño que su letra y debe cambiar proporcionalmente".

En efecto, si el tamaño de letra de un elemento aumenta hasta un valor enorme, su margen de 1em también será enorme. Si su tamaño de letra se reduce hasta un valor diminuto, el margen de 1em también será diminuto. El uso de unidades relativas permite mantener las proporciones del diseño cuando se modifica el tamaño de letra de la página.

El funcionamiento de la unidad ex es idéntico a em, salvo que en este caso, la referencia es la altura de la letra x minúscula, por lo que su valor es aproximadamente la mitad que el de la unidad em.

Por último, las medidas indicadas en píxel también se consideran relativas, ya que el aspecto de los elementos dependerá de la resolución del dispositivo en el que se visualiza la página HTML. Si un elemento tiene una anchura de 400px, ocupará la mitad de una pantalla con una resolución de 800x600, pero ocupará menos de la tercera parte en una pantalla con resolución de 1440x900.

Las unidades de medida se pueden mezclar en los diferentes elementos de una misma página, como en el siguiente ejemplo:

```
body { font-size: 10px; }
h1 { font-size: 2.5em; }
```

En primer lugar, se establece un tamaño de letra base de 10 píxel para toda la página. A continuación, se asigna un tamaño de 2.5em al elemento <h1>, por lo que su tamaño de letra real será de 2.5 x 10px = 25px.

Como se vio en los capítulos anteriores, el valor de la mayoría de propiedades CSS se hereda de padres a hijos. Así por ejemplo, si se establece el tamaño de letra al elemento <body>, todos los elementos de la página tendrán el mismo tamaño de letra, salvo que indiquen otro valor.

Sin embargo, el valor de las medidas relativas no se hereda directamente, sino que se hereda su valor real una vez calculado. El siguiente ejemplo muestra este comportamiento:

Ej03.01b-UnidadesRelativas.html:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
body {
  font-size: 12px;
  text-indent: 3em;
}
h1 { font-size: 15px }
  </style>
</head>
<body>
    <section>
      <h1>Titulo Principal <br />
    </section>
</body>
</html>
```

La propiedad text-indent, como se verá en los próximos capítulos, se utiliza para tabular la primera línea de un texto. El elemento <body> define un valor para esta propiedad, pero el elemento <h1> no lo hace, por lo que heredará el valor de su elemento padre. Sin embargo, el valor heredado no es 3em, sino 36px.

Si se heredara el valor 3em, al multiplicarlo por el valor de font-size del elemento <h1> (que vale 15px) el resultado sería 3em x 15px = 45px. No obstante, como se ha comentado, los valores que se heredan no son los relativos, sino los valores ya calculados.

Por lo tanto, en primer lugar se calcula el valor real de 3em para el elemento <body>: 3em x 12px = 36px. Una vez calculado el valor real, este es el valor que se hereda para el resto de elementos.

3.1.3 Porcentajes

El porcentaje también es una unidad de medida relativa, aunque por su importancia CSS la trata de forma separada a em, ex y px. Un porcentaje está formado por un valor numérico seguido del símbolo % y siempre está referenciado a otra medida. Cada una de las propiedades de CSS que permiten indicar como valor un porcentaje, define el valor al que hace referencia ese porcentaje.

Los porcentajes se pueden utilizar por ejemplo para establecer el valor del tamaño de letra de los elementos:

```
body { font-size: 1em; }
h1 { font-size: 200%; }
h2 { font-size: 150%; }
```

Los tamaños establecidos para los elementos <h1> y <h2> mediante las reglas anteriores, son equivalentes a 2em y 1.5em respectivamente, por lo que es más habitual definirlos mediante em.

Los porcentajes también se utilizan para establecer la anchura de los elementos:
Ej03.01c-UnidadesPorcentuales.html:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de Porcentajes</title>
  <style type="text/css">
    div#contenido { width: 600px; background-color: yellow;}
    div.principal { width: 80%; background-color: blue;}
  </style>
</head>
<body>
    <section>
      <div id="contenido">
         <h1> División General </h1>
         <div class="principal">
            <h3> División Principal </h3>
            <p> Un párrafo</p>
         </div>
            <p> Otro párrafo</p>
      </div>
    </section>
</body>
</html>
```

En el ejemplo anterior, la referencia del valor 80% es la anchura de su elemento padre. Por tanto, el elemento cuyo atributo class vale principal tiene una anchura de 80% x 600px = 480px.

3.1.4 Recomendaciones

En general, se recomienda el uso de unidades relativas siempre que sea posible, ya que mejora la accesibilidad de la página y permite que los documentos se adapten fácilmente a cualquier medio y dispositivo.

El documento **Recomendaciones sobre técnicas CSS para la mejora de la accesibilidad de los contenidos HTML**, elaborado por el organismo W3C, recomienda el uso de la unidad em para indicar el tamaño del texto y para todas las medidas que sean posibles.
El enlace al documento: http://www.w3.org/TR/WCAG10-CSS-TECHS/

Normalmente se utilizan píxel y porcentajes para definir el layout del documento (básicamente, la anchura de las columnas y de los elementos de las páginas) y em y porcentajes para el tamaño de letra de los textos.

3.2 Colores

Los colores en CSS se pueden indicar de cinco formas diferentes: palabras clave, colores del sistema, RGB hexadecimal, RGB numérico y RGB porcentual. Aunque el método más habitual es el del RGB hexadecimal, a continuación se muestran todas las alternativas que ofrece CSS.

3.2.1 Palabras clave

CSS define 147 palabras clave para referirse a los colores básicos. Las palabras se corresponden con el nombre en inglés de cada color. Son 17 básicos (aqua, black, blue, fuchsia, gray, grey, green, lime, maroon, navy, olive, purple, red, silver, teal, white y yellow) y 130 añadidos.

Ej03.02-ColoresClave.html:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
    div#contenido { width: 600px;    background-color: Aquamarine ;}
    div.principal { width: 80%;      background-color: GoldenRod ;}
  </style>
</head>
<body>
    <section>
      <div id="contenido">
         <h1> División General </h1>
         <div class="principal">
            <h3> División Principal </h3>
            <p> Un párrafo</p>
         </div>
            <p> Otro párrafo</p>
      </div>
    </section>
</body>
</html>
```

La lista ordenada alfabéticamente es esta:

Nombre Color	HEX	Color
AliceBlue	#F0F8FF	
AntiqueWhite	#FAEBD7	
Aqua	#00FFFF	
Aquamarine	#7FFFD4	
Azure	#F0FFFF	
Beige	#F5F5DC	
Bisque	#FFE4C4	
Black	#000000	
BlanchedAlmond	#FFEBCD	
Blue	#0000FF	
BlueViolet	#8A2BE2	
Brown	#A52A2A	
BurlyWood	#DEB887	
CadetBlue	#5F9EA0	
Chartreuse	#7FFF00	
Chocolate	#D2691E	
Coral	#FF7F50	
CornflowerBlue	#6495ED	
Cornsilk	#FFF8DC	
Crimson	#DC143C	
Cyan	#00FFFF	
DarkBlue	#00008B	
DarkCyan	#008B8B	
DarkGoldenRod	#B8860B	
DarkGray	#A9A9A9	
DarkGrey	#A9A9A9	
DarkGreen	#006400	
DarkKhaki	#BDB76B	
DarkMagenta	#8B008B	
DarkOliveGreen	#556B2F	
Darkorange	#FF8C00	
DarkOrchid	#9932CC	
DarkRed	#8B0000	
DarkSalmon	#E9967A	
DarkSeaGreen	#8FBC8F	
DarkSlateBlue	#483D8B	
DarkSlateGray	#2F4F4F	
DarkSlateGrey	#2F4F4F	
DarkTurquoise	#00CED1	
DarkViolet	#9400D3	
DeepPink	#FF1493	
DeepSkyBlue	#00BFFF	
DimGray	#696969	
DimGrey	#696969	

DodgerBlue	#1E90FF	
FireBrick	#B22222	
FloralWhite	#FFFAF0	
ForestGreen	#228B22	
Fuchsia	#FF00FF	
Gainsboro	#DCDCDC	
GhostWhite	#F8F8FF	
Gold	#FFD700	
GoldenRod	#DAA520	
Gray	#808080	
Grey	#808080	
Green	#008000	
GreenYellow	#ADFF2F	
HoneyDew	#F0FFF0	
HotPink	#FF69B4	
IndianRed	#CD5C5C	
Indigo	#4B0082	
Ivory	#FFFFF0	
Khaki	#F0E68C	
Lavender	#E6E6FA	
LavenderBlush	#FFF0F5	
LawnGreen	#7CFC00	
LemonChiffon	#FFFACD	
LightBlue	#ADD8E6	
LightCoral	#F08080	
LightCyan	#E0FFFF	
LightGoldenRodYellow	#FAFAD2	
LightGray	#D3D3D3	
LightGrey	#D3D3D3	
LightGreen	#90EE90	
LightPink	#FFB6C1	
LightSalmon	#FFA07A	
LightSeaGreen	#20B2AA	
LightSkyBlue	#87CEFA	
LightSlateGray	#778899	
LightSlateGrey	#778899	
LightSteelBlue	#B0C4DE	
LightYellow	#FFFFE0	
Lime	#00FF00	
LimeGreen	#32CD32	
Linen	#FAF0E6	
Magenta	#FF00FF	
Maroon	#800000	
MediumAquaMarine	#66CDAA	
MediumBlue	#0000CD	
MediumOrchid	#BA55D3	
MediumPurple	#9370DB	
MediumSeaGreen	#3CB371	
MediumSlateBlue	#7B68EE	
MediumSpringGreen	#00FA9A	
MediumTurquoise	#48D1CC	
MediumVioletRed	#C71585	

MidnightBlue	#191970	
MintCream	#F5FFFA	
MistyRose	#FFE4E1	
Moccasin	#FFE4B5	
NavajoWhite	#FFDEAD	
Navy	#000080	
OldLace	#FDF5E6	
Olive	#808000	
OliveDrab	#6B8E23	
Orange	#FFA500	
OrangeRed	#FF4500	
Orchid	#DA70D6	
PaleGoldenRod	#EEE8AA	
PaleGreen	#98FB98	
PaleTurquoise	#AFEEEE	
PaleVioletRed	#DB7093	
PapayaWhip	#FFEFD5	
PeachPuff	#FFDAB9	
Peru	#CD853F	
Pink	#FFC0CB	
Plum	#DDA0DD	
PowderBlue	#B0E0E6	
Purple	#800080	
Red	#FF0000	
RosyBrown	#BC8F8F	
RoyalBlue	#4169E1	
SaddleBrown	#8B4513	
Salmon	#FA8072	
SandyBrown	#F4A460	
SeaGreen	#2E8B57	
SeaShell	#FFF5EE	
Sienna	#A0522D	
Silver	#C0C0C0	
SkyBlue	#87CEEB	
SlateBlue	#6A5ACD	
SlateGray	#708090	
SlateGrey	#708090	
Snow	#FFFAFA	
SpringGreen	#00FF7F	
SteelBlue	#4682B4	
Tan	#D2B48C	
Teal	#008080	
Thistle	#D8BFD8	
Tomato	#FF6347	
Turquoise	#40E0D0	
Violet	#EE82EE	
Wheat	#F5DEB3	
White	#FFFFFF	
WhiteSmoke	#F5F5F5	
Yellow	#FFFF00	
YellowGreen	#9ACD32	

3.2.2 RGB decimal

En el campo del diseño gráfico, se han definido varios modelos para hacer referencia a los colores. Los dos modelos más conocidos son RGB y CMYK. Simplificando su explicación, el modelo RGB consiste en definir un color indicando la cantidad de color rojo, verde y azul que se debe mezclar para obtener ese color. Técnicamente, el modelo RGB es un modelo de tipo "aditivo", ya que los colores se obtienen sumando sus componentes.

Por lo tanto, en el modelo RGB un color se define indicando sus tres componentes R (rojo), G (verde) y B (azul). Cada una de las componentes puede tomar un valor entre cero y un valor máximo. De esta forma, el color rojo puro en RGB se crea mediante el máximo valor de la componente R y un valor de 0 para las componentes G y B.

Si todas las componentes valen 0, el color creado es el negro y si todas las componentes toman su valor máximo, el color obtenido es el blanco. En CSS, las componentes de los colores definidos mediante RGB decimal pueden tomar valores entre 0 y 255. El siguiente ejemplo establece el color del texto de un párrafo:

```
p { color: rgb(71, 98, 176); }
```

La sintaxis que se utiliza para indicar los colores es rgb() y entre paréntesis se indican las tres componentes RGB, en ese mismo orden y separadas por comas. El color del ejemplo anterior se obtendría mezclando las componentes R=71, G=98, B=176, que se corresponde con un color azul claro.

Si se indica un valor menor que 0 para una componente, automáticamente se transforma su valor en 0. Igualmente, si se indica un valor mayor que 255, se transforma automáticamente su valor a 255.

3.2.3 RGB porcentual

Las componentes RGB de un color también se pueden indicar mediante un porcentaje. El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal. La única diferencia es que en este caso el valor de las componentes RGB puede tomar valores entre 0% y 100%. Por tanto, para transformar un valor RGB decimal en un valor RGB porcentual, es preciso realizar una regla de tres considerando que 0 es igual a 0% y 255 es igual a 100%.

El mismo color del ejemplo anterior se puede representar de forma porcentual:

```
p { color: rgb(27%, 38%, 69%); }
```

Al igual que sucede con el RGB decimal, si se indica un valor inferior a 0%, se transforma automáticamente en 0% y si se indica un valor superior a 100%, se trunca su valor a 100%.

3.2.4 RGB hexadecimal

Aunque es el método más complicado para indicar los colores, se trata del método más utilizado con mucha diferencia. De hecho, prácticamente todos los sitios web reales utilizan exclusivamente este método.

Para entender el modelo RGB hexadecimal, en primer lugar es preciso introducir un concepto matemático llamado sistema numérico hexadecimal. Cuando realizamos operaciones matemáticas, siempre utilizamos 10 símbolos para representar los números (del 0 al 9). Por este motivo, se dice que utilizamos un sistema numérico decimal.

No obstante, el sistema decimal es solamente uno de los muchos sistemas numéricos que se han definido. Entre los sistemas numéricos alternativos más utilizados se encuentra el sistema hexadecimal, que utiliza 16 símbolos para representar sus números.

Como sólo conocemos 10 símbolos numéricos, el sistema hexadecimal utiliza también seis letras (de la A a la F) para representar los números. De esta forma, en el sistema hexadecimal, después del 9 no va el 10, sino la A. La letra B equivale al número 11, la C al 12, la D al 13, la E al 14 y la F al número 15.

Definir un color en CSS con el método RGB hexadecimal requiere realizar los siguientes pasos: - Determinar las componentes RGB decimales del color original, por ejemplo: R = 71, G = 98, B = 176 - Transformar el valor decimal de cada componente al sistema numérico hexadecimal. Se trata de una operación exclusivamente matemática, por lo que puedes utilizar una calculadora. En el ejemplo anterior, el valor hexadecimal de cada componente es: R = 47, G = 62, B = B0 - Para obtener el color completo en formato RGB hexadecimal, se concatenan los valores hexadecimales de las componentes RGB en ese orden y se les añade el prefijo #. De esta forma, el color del ejemplo anterior es #4762B0 en formato RGB hexadecimal.

Siguiendo el mismo ejemplo de las secciones anteriores, el color del párrafo se indica de la siguiente forma utilizando el formato RGB hexadecimal:

p { color: #4762B0; }

Recuerda que aunque es el método más complicado para definir un color, se trata del método que utilizan la inmensa mayoría de sitios web, por lo que es imprescindible dominarlo. Afortunadamente, todos los programas de diseño gráfico convierten de forma automática los valores RGB decimales a sus valores RGB hexadecimales, por lo que no tienes que hacer ninguna operación matemática.

El formato RGB hexadecimal es la forma más compacta de indicar un color, ya que incluso es posible comprimir sus valores cuando todas sus componentes son iguales dos a dos:

```
#AAA = #AAAAAA
#FFF = #FFFFFF
#A0F = #AA00FF
#369 = #336699
```

En el siguiente ejemplo se establece el color de fondo de la página a blanco, el color del texto a negro y el color de la letra de los titulares se define de color rojo:

```
body { background-color: #FFF; color: #000; }
h1, h2, h3, h4, h5, h6 { color: #C00; }
```

Las letras que forman parte del color en formato RGB hexadecimal se pueden escribir en mayúsculas o minúsculas indistintamente. No obstante, se recomienda escribirlas siempre en mayúsculas o siempre en minúsculas para que la hoja de estilos resultante sea más limpia y homogénea.

3.2.5 Colores del sistema

Los colores del sistema son similares a los colores indicados mediante su nombre, pero en este caso hacen referencia al color que muestran algunos elementos del sistema operativo del usuario.

Existen varios colores definidos, como por ejemplo ActiveBorder, que hace referencia al color del borde de las ventanas activas. Los colores del sistema son los siguientes:

ActiveBorder	Borde de la ventana activa.
ActiveCaption	Título de la ventana activa.
AppWorkspace	Color de fondo de interfaz de documentos múltiples
Background	Fondo del escritorio.
ButtonFace	Color Frontal para los elementos tridimensionales de la pantalla.
ButtonHighlight	Resalte color para los elementos tridimensionales de pantalla (para los bordes más alejados de la fuente de luz).
ButtonShadow	Color de sombra para los elementos tridimensionales de la pantalla.
ButtonText	Texto de Botones
CaptionText	Texto del título de tablas, tamaño de la caja, y la caja de flecha de desplazamiento.
GrayText	Texto grisáceo. Este color se ajusta a # 000 si el controlador de pantalla actual no es compatible con un color gris sólido.
Highlight	Artículo (s) seleccionado en un control.
HighlightText	Texto del artículo (s) seleccionado en un control
InactiveBorder	Borde de la ventana inactiva.
InactiveCaption	Título de la ventana inactiva.
InactiveCaptionText	Color del texto de un título inactivo.
InfoBackground	Color de fondo para los controles de información sobre herramientas.
InfoText	Color del texto de los controles sobre herramientas.
Menu	Fondo del menú.
MenuText	Texto del menú
Scrollbar	Barra de desplazamiento
ThreeDDarkShadow	Sombra oscura de los elementos tridimensionales de pantalla.
ThreeDFace	Color Frontal para los elementos tridimensionales de la pantalla.
ThreeDHighlight	Resalte color para los elementos tridimensionales de la pantalla
ThreeDLightShadow	Color de la luz de los elementos tridimensionales de pantalla (para los bordes que se enfrenta la fuente de luz).
ThreeDShadow	Sombra oscura de los elementos tridimensionales de pantalla.
Window	Ventana fondo.
WindowFrame	Marco de ventana.
WindowText	Texto en ventanas.

Aunque es posible definir los colores en CSS utilizando estos nombres, se trata de un método que nunca se utiliza, por lo que se puede considerar prácticamente como una rareza de CSS.

3.2.6 Colores web safe.

Como cada componente RGB de los colores puede tomar un valor entre 0 y 255, el número total de colores que se pueden representar con este formato es de 256 x 256 x 256 = 16.777.216 colores. Sin embargo, en la década de los 90 los monitores de los usuarios no eran capaces de mostrar más de 256 colores diferentes.

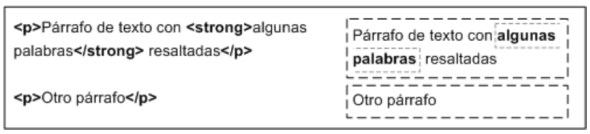
A partir de todos los colores disponibles, se eligieron 216 colores que formaron la paleta de colores "web safe". Esta paleta de colores podía ser utilizada por los diseñadores con la seguridad de que se verían correctamente en cualquier navegador de cualquier sistema operativo.

Hoy en día, su importancia ha descendido notablemente, ya que prácticamente todos los usuarios utilizan dispositivos con una profundidad de color de 16 y 32 bits. No obstante, el auge en el uso de los dispositivos móviles hace que siga siendo un tema a considerar, ya que las pantallas de muchos móviles sólo pueden representar un número reducido de colores.
La lista completa de colores web safe y sus valores hexadecimales.

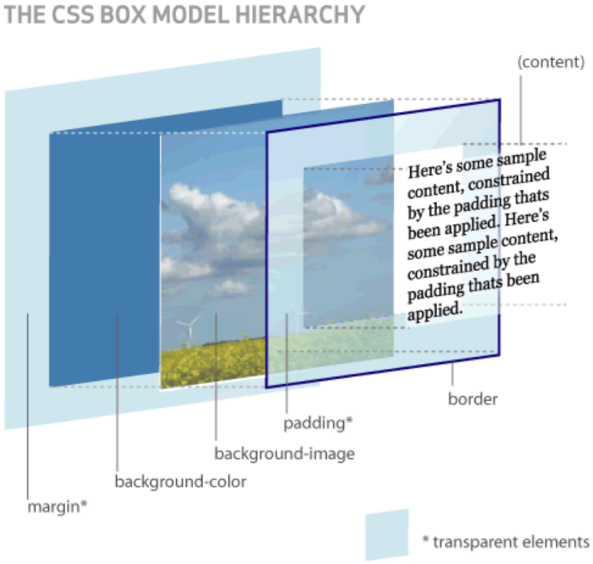
000	300	600	900	C00	*F00*
003	303	603	903	C03	*F03*
006	306	606	906	C06	F06
009	309	609	909	C09	F09
00C	30C	60C	90C	C0C	F0C
00F	30F	60F	90F	C0F	*F0F*
030	330	630	930	C30	F30
033	333	633	933	C33	F33
036	336	636	936	C36	F36
039	339	639	939	C39	F39
03C	33C	63C	93C	C3C	F3C
03F	33F	63F	93F	C3F	F3F
060	360	660	960	C60	F60
063	363	663	963	C63	F63
066	366	666	966	C66	F66
069	369	669	969	C69	F69
06C	36C	66C	96C	C6C	F6C
06F	36F	66F	96F	C6F	F6F
090	390	690	990	C90	F90
093	393	693	993	C93	F93
096	396	696	996	C96	F96
099	399	699	999	C99	F99
09C	39C	69C	99C	C9C	F9C
09F	39F	69F	99F	C9F	F9F
0C0	3C0	6C0	9C0	CC0	FC0
0C3	3C3	6C3	9C3	CC3	FC3
0C6	3C6	6C6	9C6	CC6	FC6
0C9	3C9	6C9	9C9	CC9	FC9
0CC	3CC	6CC	9CC	CCC	FCC
0CF	3CF	6CF	9CF	CCF	FCF
0F0	3F0	*6F0*	9F0	CF0	*FF0*
0F3	*3F3*	*6F3*	9F3	CF3	*FF3*
0F6	*3F6*	6F6	9F6	*CF6*	*FF6*
0F9	3F9	6F9	9F9	CF9	FF9
0FC	*3FC*	6FC	9FC	CFC	FFC
0FF	*3FF*	*6FF*	9FF	CFF	*FFF*

4 MODELO DE CAJAS

El modelo de cajas o "box model" es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El modelo de cajas es el comportamiento de CSS que hace que todos los elementos de las páginas se representen mediante cajas rectangulares.



Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento. La imagen superior muestra las tres cajas rectangulares que crean las tres etiquetas HTML que incluye la página.



Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características.

Cada una de las cajas está formada por seis partes, tal y como muestra la siguiente imagen de la izquierda.
(Ref: http://www.hicksdesign.co.uk/boxmodel/)

Las partes que componen cada caja y su orden de visualización desde el punto de vista del usuario son las siguientes:

Contenido (content): se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)

Relleno (padding): espacio libre opcional existente entre el contenido y el borde.
Borde (border): línea que encierra completamente el contenido y su relleno.
Imagen de fondo (background image): imagen que se muestra por detrás del contenido y el espacio de relleno.
Color de fondo (background color): color que se muestra por detrás del contenido y el espacio de relleno.
Margen (margin): separación opcional existente entre la caja y el resto de cajas adyacentes.

El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos) y en el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos). Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos).

Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza. No obstante, si la imagen de fondo no cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo. Combinando imagenes transparentes y colores de fondo se pueden lograr efectos gráficos muy interesantes.

4.1 Anchura y altura

4.1.1 Anchura

La propiedad CSS que controla la anchura de la caja de los elementos se denomina width.

Propiedad	**width**
Valores	<medida> | <porcentaje> | auto | inherit
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las filas de tabla y los grupos de filas de tabla
Valor inicial	auto
Descripción	Establece la anchura de un elemento

La propiedad width no admite valores negativos y los valores en porcentaje se calculan a partir de la anchura de su elemento padre. El valor inherit indica que la anchura del elemento se hereda de su elemento padre. El valor auto, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la anchura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página. CSS define otras dos propiedades relacionadas con la anchura de los elementos: min-width y max-width, que se verán más adelante.

4.1.2 Altura

La propiedad CSS que controla la altura de los elementos se denomina height.

Propiedad	**height**
Valores	<medida> | <porcentaje> | auto | inherit
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las columnas de tabla y los grupos de columnas de tabla
Valor inicial	auto
Descripción	Establece la altura de un elemento

Al igual que sucede con width, la propiedad height no admite valores negativos. Si se indica un porcentaje, se toma como referencia la altura del elemento padre. Si el elemento padre no tiene una altura definida explícitamente, se asigna el valor auto a la altura.

El valor inherit indica que la altura del elemento se hereda de su elemento padre. El valor auto, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la altura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página. CSS define otras dos propiedades relacionadas con la altura de los elementos: min-height y max-height, que se verán más adelante.

Veamos un ejemplo con **width** y **height:** **`Ej04.01a-HeightWidth.html`**:

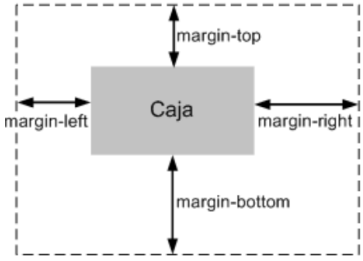
```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de Width y Height</title>
  <style type="text/css">
    #pixeles { height: 300px; width: 200px; background-color: blue;}
    #porcentaje { height: 30%; width: 20%; background-color: red;
    position: absolute; top: 400px;}
  </style>
</head>
<body>
    <section>
      <div id="pixeles"></div>
      <div id="porcentaje"></div>
    </section>
</body> </html>
```

4.2 Margen y relleno

4.2.1 Margen

CSS define cuatro propiedades para controlar cada uno de los márgenes horizontales y verticales de un elemento.

Propiedades	**margin-top, margin-right, margin-bottom, margin-left**
Valores	<medida> | <porcentaje> | auto | inherit
Se aplica a	Todos los elementos, salvo margin-top y margin-bottom que sólo se aplican a los elementos de bloque y a las imágenes
Valor inicial	0
Descripción	Establece cada uno de los márgenes horizontales y verticales de un elemento



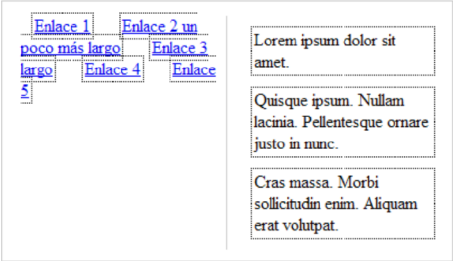
Cada una de las propiedades establece la separación entre el borde lateral de la caja y el resto de cajas adyacentes:

Las unidades más utilizadas para indicar los márgenes de un elemento son los píxeles (cuando se requiere una precisión total), los em (para hacer diseños que mantengan las proporciones) y los porcentajes (para hacer diseños líquidos o fluidos).

El siguiente ejemplo añade márgenes al segundo párrafo **`Ej04.02a-Margenes.html`**:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
    .destacado { margin-left: 2em; margin-right: 200px;}
  </style>
</head>
<body>
    <section>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nam et elit.
      Vivamus placerat lorem. Maecenas sapien. Integer ut massa. Cras diam ipsum,
      laoreet non, tincidunt a, viverra sed, tortor.</p>

      <p class="destacado">Vestibulum lectus diam, luctus vel, venenatis ultrices,
      cursus vel, tellus. Etiam placerat erat non sem. Nulla molestie odio non
      nisl tincidunt faucibus.</p>

      <p>Aliquam euismod sapien eu libero. Ut tempor orci at nulla. Nam in eros
      egestas massa vehicula nonummy. Morbi posuere, nibh ultricies consectetuer tincidunt,
      risus turpis laoreet elit, ut tincidunt risus sem et nunc.</p>
    </section>
</body>
</html>
```

Algunos diseñadores web utilizan la etiqueta <blockquote> para tabular los contenidos de los párrafos. Se trata de un error grave porque HTML no debe utilizarse para controlar el aspecto de los elementos. CSS es el único responsable de establecer el estilo de los elementos, por lo que en vez de utilizar la etiqueta <blockquote> de HTML, debería utilizarse la propiedad margin-left de CSS.



Los márgenes verticales (margin-top y margin-bottom) sólo se pueden aplicar a los elementos de bloque y las imágenes, mientras que los márgenes laterales (margin-left y margin-right) se pueden aplicar a cualquier elemento, tal y como muestra la siguiente imagen:

La imagen muestra el resultado de aplicar los mismos márgenes a varios enlaces (elementos en línea) y varios párrafos (elementos de bloque). En los elementos en línea los márgenes verticales no tienen ningún efecto, por lo que los enlaces no muestran ninguna separación vertical, al contrario de lo que sucede con los párrafos. Sin embargo, los márgenes laterales funcionan sobre cualquier tipo de elemento, por lo que los enlaces se muestran separados entre sí y los párrafos aumentan su separación con los bordes laterales de su elemento contenedor.

Además de las cuatro propiedades que controlan cada uno de los márgenes del elemento, CSS define una propiedad especial que permite establecer los cuatro márgenes de forma simultánea. Estas propiedades especiales se denominan "propiedades shorthand" y CSS define varias propiedades de este tipo, como se verá más adelante.

La propiedad que permite definir de forma simultanea los cuatro márgenes se denomina margin.

Propiedad	**margin**
Valores	(<medida> | <porcentaje> | auto) {1, 4} | inherit
Se aplica a	Todos los elementos salvo algunos casos especiales de elementos mostrados como tablas
Valor inicial	-
Descripción	Establece de forma directa todos los márgenes de un elemento

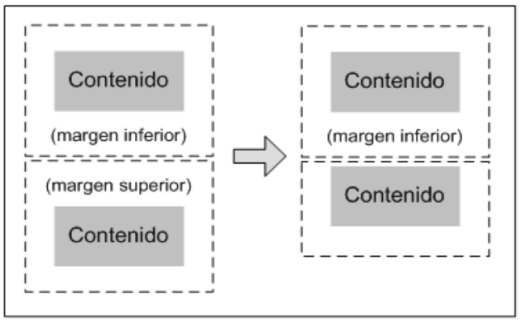
La notación {1, 4} de la definición anterior significa que la propiedad margin admite entre uno y cuatro valores, con el siguiente significado:

- Si solo se indica un valor, todos los márgenes tienen ese valor.
- Si se indican dos valores, el primero se asigna al margen superior e inferior y el segundo se asigna a los márgenes izquierdo y derecho.
- Si se indican tres valores, el primero se asigna al margen superior, el tercero se asigna al margen inferior y el segundo valor se asigna los márgenes izquierdo y derecho.
- Si se indican los cuatro valores, el orden de asignación es: margen superior, margen derecho, margen inferior y margen izquierdo.

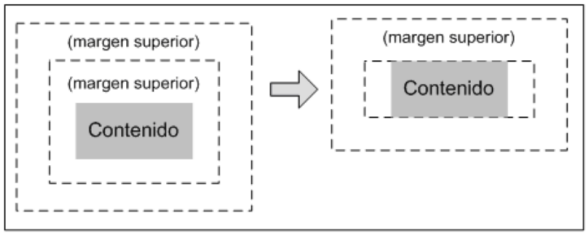
Otro ejemplo (modificamos el ejemplo anterior): **Ej04.02b-OtrosMargenes.html**:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
  <style type="text/css">
    .destacado { margin: 30px 2em 10% 20ex;}
  </style>
</head>
<body>
    <section>
      <p>Lorem ipsum dolor ...</p>
      <p class="destacado">Vestibulum lectus diam, ...</p>
      <p>Aliquam euismod sapien eu libero...</p>
    </section>
</body>
</html>
```

El comportamiento de los márgenes verticales es más complejo de lo que se puede imaginar. Cuando se juntan dos o más márgenes verticales, se fusionan de forma automática y la altura del nuevo margen será igual a la altura del margen más alto de los que se han fusionado.



De la misma forma, si un elemento está contenido dentro de otro elemento, sus márgenes verticales se fusionan y resultan en un nuevo margen de la misma altura que el mayor margen de los que se han fusionado:



Aunque en principio puede parecer un comportamiento extraño, la razón por la que se propuso este mecanismo de fusión automática de márgenes verticales es el de dar uniformidad a las páginas web habituales. En una página con varios párrafos, si no se diera este comportamiento y se estableciera un determinado margen a todos los párrafos, el primer párrafo no mostraría un aspecto homogéneo respecto de los demás.

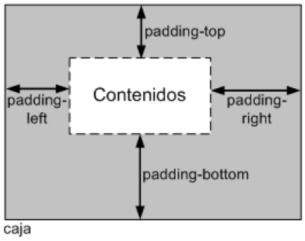
En el caso de un elemento que se encuentra en el interior de otro y sus márgenes se fusionan de forma automática, se puede evitar este comportamiento añadiendo un pequeño relleno (padding: 1px) o un borde (border: 1px solid transparent) al elemento contenedor.

4.2.2 Relleno

CSS define cuatro propiedades para controlar cada uno de los espacios de relleno horizontales y verticales de un elemento.

Propiedades	**padding-top, padding-right, padding-bottom, padding-left**
Valores	<medida> | <porcentaje> | inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	0
Descripción	Establece cada uno de los rellenos horizontales y verticales de un elemento

Cada una de estas propiedades establece la separación entre el contenido y los bordes laterales de la caja del elemento:



Como sucede con los márgenes, CSS también define una propiedad de tipo "shorthand" llamada padding para establecer los cuatro rellenos de un elemento de forma simultánea.

Propiedad	**padding**
Valores	(<medida> | <porcentaje>) {1, 4} | inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	-
Descripción	Establece de forma directa todos los rellenos de los elementos

La notación {1, 4} de la definición anterior significa que la propiedad padding admite entre uno y cuatro valores, con el mismo significado que el de la propiedad margin. Ejemplo:

```
body {padding: 2em}     /* Todos los rellenos valen 2em */
body {padding: 1em 2em} /* Superior e inferior = 1em, Izquierdo y derecho = 2em */

/* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 2em */
body {padding: 1em 2em 3em}

/* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 4em */
body {padding: 1em 2em 3em 4em}
```

Ejercicio 3

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir los siguientes márgenes y rellenos. La página original es similar a esta:

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Ejercicio 3</title>
   <link href="css/Ejercicio03a.css" rel="stylesheet" />
</head>
<body>
<section>
<!--Inicio Contenedor -->
<div id="contenedor">

      <div id="cabecera">  <!--inicio cabecera-->
            <div id="logo">
               <h1> <span>Mis Noticias</span></h1>
            </div>
            <div id="buscador">
                  <form action="php/Ejercicio03.php" method="post">
                  Buscar: <input type="search" placeholder="realice su busqueda" />
                  </form>
            </div>
            <div class="clear"></div>
      </div>
      <!--fin cabecera-->
```

```
<!--inicio menu principal-->
<div id="menu">
      <ul>
            <li><a href="#"> enviar historia</a> |</li>
            <li><a href="#"> portada</a> |</li>
            <li><a href="#"> pendientes</a> |</li>
            <li><a href="#"> populares</a> |</li>
            <li><a href="#"> más visitadas</a> |</li>
            <li><a href="#"> destacadas</a> |</li>
      </ul>
      <div class="clear"></div>
</div>
<!--fin menu principal-->

<!--Inicio Lateral Izquierdo -->
<div id="lateral">

      <!--Inicio Sección Noticias. Enlaces -->
      <div id="noticias">
            <h3><a href="#">Noticias</a></h3>
            <p><span class="fecha">dd/mm/aaaa</span></p>
            <p><a href="#">Lorem ipsum dolor</a></p>
            <p><span class="fecha">dd/mm/aaaa</span></p>
            <p><a href="#">Lorem ipsum dolor</a></p>
            <p><span class="fecha">dd/mm/aaaa</span></p>
            <p><a href="#">Lorem ipsum dolor</a></p>
            <p><span class="fecha">dd/mm/aaaa</span></p>
            <p><a href="#">Lorem ipsum dolor</a></p>
            <p><span class="fecha">dd/mm/aaaa</span></p>
            <p><a href="#">Lorem ipsum dolor</a></p>

            <br />

            <h3><a href="#">Enlaces relacionados</a></h3>
            <ul>
                  <li><a href="#"> ipsum</a></li>
                  <li><a href="#"> dolor</a></li>
                  <li><a href="#"> sit</a></li>
            </ul>
      </div>
      <!--Fin Sección Noticias. Enlaces -->

      <!-- Inicio Sección Publicidad-->
      <div id="publicidad">
            <h3><a href="#">Publicidad</a></h3>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
            <p>Vivamus quis arcu massa.</p>
            <p>Etiam nec risus sit amet lorem tempor hendrerit.</p>
            <p><a href="#">Seguir leyendo...</a></p>
      </div>
      <!-- Fin Sección Publicidad-->

</div>
<!-- Fin Lateral Izquierdo-->

<!-- Inicio Contenido-->
<div id="contenido">

      <!-- Inicio Cuerpo Principal-->
      <div id="principal">

            <!-- Inicio Artículo 1 -->
            <div class="articulo">
         <h3>El precio de la libertad de Android  </h3>
         <img src="img/android.jpg" alt="Mi Futuro móvil" />
         <p>Que Android es un sistema operativo libre (en el sentido de Open
         Source, se entiende) es algo que estamos hartos de oir. Sin embargo a veces
         hay ciertas noticias que nos hacen dudar que esta libertad sea tal como en
         un principio puede parecer.
```

```
                Puesto que la principal diferencia de base de Android con respecto a sus
                competidores es ser 'libre', creemos que este tema merece una buena
                explicación para aclarar todo lo que hay detrás de esta supuesta
                libertad.</p>
                        <p>
<a href="http://www.elandroidelibre.com/2013/01/el-precio-de-la-libertad-de-android.html">
                      Seguir leyendo...</a></p>

                  </div>


                  <!-- Fin Artículo 1 -->

                  <!-- Inicio Artículo 2 -->
                  <div class="articulo">
                  <h3>Richard Stallman:
                  "Tener el control de tu informática es un derecho humano" </h3>
                  <img src="img/stallman.jpg" alt="Dios Todopoderoso" />
                  <p>"Ecuador es el mejor ejemplo de una política estatal de migración a
                  software libre de las agencias públicas. Con el software libre los
                  usuarios tienen el control del programa, si éste no es libre, es el
                  programa quien controla a los usuarios. Cuando el usuario es el Estado,
                  estamos hablando del control de su soberanía informática. Cualquier país
                  debe migrar al software libre para echar al software privativo, ahora
                  mismo Ecuador tiene la mejor política en ese sentido." </p>
                  <p>
<a href="http://www.lamarea.com/2013/01/30/tener-el-control-de-tu-informatica-es-un-
derecho-humano/">Seguir leyendo...</a></p>

                  </div>
                  <!-- Fin Artículo 2 -->

            </div>
            <!-- Fin Cuerpo Principal-->

            <!-- Inicio Cuerpo Secundario-->
            <div id="secundario">
                  <h3>Las mejores catástrofes culinarias de Pinterest</h3>
                  <p>La vida no es como aparece en Pinterest. Y la cocina, menos.
                  Y la repostería, menos todavía. </p>
                  <p><a href="#">Seguir leyendo...</a></p>

                  <br />

                  <h3>La madre que usó una tarjeta perdida para comprar pañales,
                  a prisión en 15 días</h3>
                  <p>Emilia Soria, la joven que hace seis años utilizó una tarjeta
                  que se encontró en la calle para comprar pañales y comida para sus hijas,
                  irá a la cárcel en un plazo máximo de quince días.
                  </p>
                  <p><a href="#">Seguir leyendo...</a></p>

            </div>
            <!-- Fin Cuerpo Secundario-->

      </div>
      <!-- Fin Contenido -->

      <div class="clear"></div>
```

```
      <!--Inicio Pie -->
      <div id="pie">
            <span class="enlaces">
                  <a href="#"> todas</a> |
                  <a href="#"> actualidad</a> |
                  <a href="#"> cultura</a> |
                  <a href="#"> ocio</a> |
                  <a href="#"> tecnología</a>

            </span>
            <span class="copyright">
                  &copy; 2013 Iván Rodríguez
            </span>
            <div class="clear"></div>

      </div>
      <!--Fin Pie -->

</div>
<!--Fin Contenedor -->

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

<!-- AHORA EL CSS, Ejercicio03.css →

```
/* ------- CONTENEDOR ---------*/
#contenedor {
   width: 90%;
   max-width: 900px;
   margin:0 auto;

   /* La URL de la textura es: (buscamos en Google fondos web blue)
   www.miswallpapers.net/2008-2009Walls/varios/wallpaper_97_Blue_lines_1600x1200.jpg */
   background-image: url('../img/textura.jpg');
}

/* ################## CAJAS PRINCIPALES #################### */
#cabecera, #menu, #lateral, #contenido, #principal, #secundario, #pie {
   border: 2px solid #777;
}

#noticias, #publicidad, #principal, #secundario {}

#cabecera, #menu {    clear: both;   }


/* ---------  CABECERA ---------*/
#cabecera { }

#logo{   float:left;   }

#buscador{  float: right;     }


/* -------- MENU SUPERIOR -----------*/
#menu {}

#menu ul, #menu li{
   display: inline;
   float:left;
}
```

```
/* -------- CAJA LATERAL ----------- */

#lateral {
   float: left;
   width: 20%;
}

#publicidad {
}


/* ############ CAJA CONTENIDO ############# */

#contenido {
   float: right;
   width: 78%;
}

/* --------- CAJA PRINCIPAL --------*/
#principal {
   float: left;
   width: 68%;
}

img{  border: none;   }

.articulo img{
   width: 100px;
   float: left;
}

.articulo {    font-size: 15px;      }


/* -------------- COLUMNA DERECHA -----------*/
#secundario {
   float: right; /*se ha cambiado de left a right*/
   width: 25%;
}

/* ############ FIN CAJA CONTENIDO ############# */


/* ------------ CAJA PIE ------------*/
#pie{ clear: both;    }

#pie .enlaces{ float:left;    }

#pie .copyright{      float: right;  }


/* ######## REGLAS GENÉRICAS ####### */

ul, ul li{
   margin: 0;
   padding: 0;
   list-style: none;
}


h1, h3, p, form{
   margin: 0;
   padding: 0;
}

.clear { clear: both; }
```

Se pide:

1. El elemento #cabecera debe tener un relleno de 1em en todos los lados.
2. El elemento #menu debe tener un relleno de 0.5em en todos los lados y un margen inferior y superior de 0.5em.
3. El resto de elementos (#noticias, #publicidad, #principal, #secundario) deben tener 0.1em de relleno en todos sus lados, salvo el elemento #pie, que sólo debe tener relleno en la zona superior e inferior.
4. Los elementos .articulo deben mostrar una separación entre ellos de 1em.
5. Las imágenes de los artículos muestran un margen de 0.5em en todos sus lados.
6. El elemento #publicidad está separado 1em de su elemento superior.
7. El elemento #pie debe tener un margen superior de 1em.

Una vez finalizado el resultado será IGUAL a este:

4.3 Bordes

CSS permite modificar el aspecto de cada uno de los cuatro bordes de la caja de un elemento. Para cada borde se puede establecer su anchura o grosor, su color y su estilo, por lo que en total CSS define 20 propiedades relacionadas con los bordes.

4.3.1 Anchura

La anchura de los bordes se controla con las cuatro propiedades siguientes:

Propiedades	**border-top-width, border-right-width, border-bottom-width, border-left-width**
Valores	(<medida> | thin | medium | thick) | inherit
Se aplica a	Todos los elementos
Valor inicial	Medium
Descripción	Establece la anchura de cada uno de los cuatro bordes de los elementos

La anchura de los bordes se indica mediante una medida (en cualquier unidad de medida absoluta o relativa) o mediante las palabras clave thin (borde delgado), medium (borde normal) y thick (borde ancho).

La unidad de medida más habitual para establecer el grosor de los bordes es el píxel, ya que es la que permite un control más preciso sobre el grosor. Las palabras clave apenas se utilizan, ya que el estándar CSS no indica explícitamente el grosor al que equivale cada palabra clave, por lo que pueden producirse diferencias visuales entre navegadores. Así por ejemplo, el grosor medium equivale a 4px en algunas versiones de Internet Explorer y a 3px en el resto de navegadores.

Ej04.03a-AnchuraBorde.html:

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Anchura con CSS</title>

  <style type="text/css">
    div {
     border:2px solid; /* Aplicamos 1° un borde */
     border-top-width: 10px;  /* Sin bordes no podremos ver los anchos ;D */
     border-right-width: 1em;
     border-bottom-width: thick;
     border-left-width: thin;
  </style>
</head>

<body>
    <section>
      <div>
         <p>Lorem ipsum dolor ...</p>
         <p>Vestibulum lectus diam, ...</p>
         <p>Aliquam euismod sapien eu libero...</p>
      </div>
    </section>
</body>
</html>
```

NOTA IMPORTANTE: Para poder visualizar los anchos de los bordes previamente hay que definir dichos bordes mediante la propiedad border.
NOTA2: Para dejar parte del código como comentario, hay que seleccionarlo y pulsar CTRL + 3;
NOTA3: En el caso de usar direcciones URL muy largas podemos usar acortadores como http://goo.gl/

Si se quiere establecer de forma simultánea la anchura de todos los bordes de una caja, es necesario utilizar una propiedad "shorthand" llamada border-width:

Propiedad	**border-width**
Valores	(<medida> | thin | medium | thick) {1, 4} | inherit
Se aplica a	Todos los elementos
Valor inicial	Medium
Descripción	Establece la anchura de todos los bordes del elemento

La propiedad border-width permite indicar entre uno y cuatro valores. El significado de cada caso es el habitual de las propiedades "shorthand":

```
p { border-width: thin }                    /* thin thin thin thin */
p { border-width: thin thick }              /* thin thick thin thick */
p { border-width: thin thick medium }       /* thin thick medium thick */
p { border-width: thin thick medium thin } /* thin thick medium thin */
```

Si se indica un solo valor, se aplica a los cuatro bordes. Si se indican dos valores, el primero se aplica al borde superior e inferior y el segundo valor se aplica al borde izquierdo y derecho.

Si se indican tres valores, el primero se aplica al borde superior, el segundo se aplica al borde izquierdo y derecho y el tercer valor se aplica al borde inferior. Si se indican los cuatro valores, el orden de aplicación es superior, derecho, inferior e izquierdo.

4.3.2 Color

El color de los bordes se controla con las cuatro propiedades siguientes:

Propiedades	**border-top-color, border-right-color, border-bottom-color, border-left-color**
Valores	<color> | transparent | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de cada uno de los cuatro bordes de los elementos

Modificamos el ejemplo anterior: `Ej04.03b-ColorBorde.html`:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Anchura con CSS</title>
  <style type="text/css">
    div {
      border:2px solid;
      border-width: 10px 1em thick thin; /* Hemos hecho un SHORTHAND*/
      /* NOTA IMPORTANTE:
      No queda igual un borde de 0px que un borde de por ej. 5px y color transparent */
      border-top-color: #CC0000;
      border-right-color: blue;
      border-bottom-color: #00FF00;
      border-left-color: #CCC;  }
  </style>
</head>
<body>
    <section>
      <div>
   <p>Lorem ipsum dolor ...</p>
   <p>Vestibulum lectus diam, ...</p>
   <p>Aliquam euismod sapien eu libero...</p>
      </div>
    </section> </body>
</html>
```

CSS incluye una propiedad "shorthand" llamada border-color para establecer de forma simultánea el color de todos los bordes de una caja:

Propiedad	**border-color**
Valores	(<color> | transparent) {1, 4} | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de todos los bordes del elemento

En este caso, al igual que sucede con la propiedad border-width, es posible indicar de uno a cuatro valores y las reglas de aplicación son idénticas a las de la propiedad border-width.

4.3.3 Estilo de Bordes

CSS permite establecer el estilo de cada uno de los bordes mediante las siguientes propiedades:

Propiedades	**border-top-style, border-right-style, border-bottom-style, border-left-style**
Valores	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el estilo de cada uno de los cuatro bordes de los elementos

El estilo de los bordes sólo se puede indicar mediante alguna de las palabras reservadas definidas por CSS. Como el valor por defecto de esta propiedad es none, los elementos no muestran ningún borde visible a menos que se establezca explícitamente un estilo de borde.
Para ver cómo queda, lo mejor es verlo con un ejemplo **Ej04.03c-EstiloBorde.html**::

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Anchura con CSS</title>
  <style type="text/css">
       p { border-width: 4px; }
       .a { border-style: dashed}
       .b { border-style: dotted}
       .c { border-style: double}
       .d { border-style: groove}
       .e { border-style: inset}
       .f { border-style: outset}
       .g { border-style: ridge}
       .h { border-style: solid}
       .especial {    border-color: red;
                      border-top-style: dashed;     border-right-style: ridge;
                      border-bottom-style: groove;  border-left-style: solid;     }
  </style>
</head>
<body>
    <section>
      <div>
        <p class="a">Lorem ipsum dolor ...</p>
        <p class="b">Lorem ipsum dolor ...</p>
        <p class="c">Lorem ipsum dolor ...</p>
        <p class="d">Lorem ipsum dolor ...</p>
        <p class="e">Lorem ipsum dolor ...</p>
        <p class="f">Lorem ipsum dolor ...</p>
        <p class="g">Lorem ipsum dolor ...</p>
        <p class="h">Lorem ipsum dolor ...</p>
        <p class="especial"> Estilos Múltiples</p>
      </div>
    </section>
</body>
</html>
```

Los bordes más utilizados son solid y dashed, seguidos de double y dotted. Los estilos none y hidden son idénticos visualmente, pero se diferencian en la forma que los navegadores resuelven los conflictos entre los bordes de las celdas adyacentes en las tablas.

Para establecer de forma simultánea los estilos de todos los bordes de una caja, es necesario utilizar la propiedad "shorthand" llamada border-style:

Propiedad	**border-style**
Valores	(none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset) {1, 4} | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo de todos los bordes del elemento

Como es habitual, la propiedad permite indicar de uno a cuatro valores diferentes y las reglas de aplicación son las habituales de las propiedades "shorthand".

4.3.4 Propiedades shorthand

Como sucede con los márgenes y los rellenos, CSS define una serie de propiedades de tipo "shorthand" que permiten establecer todos los atributos de los bordes de forma simultánea. CSS incluye una propiedad "shorthand" para cada uno de los cuatro bordes y una propiedad "shorthand" global.

Propiedades	**border-top, border-right, border-bottom, border-left**
Valores	(<medida_borde> || <color_borde> || <estilo_borde>) | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de cada uno de los cuatro bordes de los elementos

Las propiedades "shorthand" permiten establecer alguno o todos los atributos de cada borde. El siguiente ejemplo establece el color y el tipo del borde inferior, pero no su anchura:

```
h1 {
  border-bottom: solid red;
}
```

En el ejemplo anterior, la anchura del borde será la correspondiente al valor por defecto (medium). Este otro ejemplo muestra la forma habitual utilizada para establecer el estilo de cada borde:

```
div {
  border-top: 1px solid #369;
  border-bottom: 3px double #369;
}
```

Veamos un ejemplo con lo anterior: **Ej04.03d-BordesH1Div.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Bordes</title>
  <style type="text/css">
    h1 { border-bottom: solid red; }
   div {    border-top: 1px solid #369;      border-bottom: 3px double #369;      }
  </style>
</head>
<body>
    <section>
      <h1> Ejemplos de Bordes</h1>
      <div> <p >Lorem ipsum dolor ...</p></div>
    </section> </body>
</html>
```

Por ultimo, CSS define una propiedad de tipo *"shorthand"* global para establecer el valor de todos los atributos de todos los bordes de forma directa:

Propiedad	**border**
Valores	(<medida_borde> || <color_borde> || <estilo_borde>) | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de todos los bordes de los elementos

Las siguientes reglas CSS son equivalentes:

```
div {  border-top: 1px solid red;
       border-right: 1px solid red;
       border-bottom: 1px solid red;
       border-left: 1px solid red;
}
div { border: 1px solid red; }
```

Como el valor por defecto de la propiedad border-style es none, si una propiedad shorthand no establece explícitamente el estilo de un borde, el elemento no muestra ese borde:

```
/* Sólo se establece el color, por lo que el estilo es
    "none" y el borde no se muestra */
div { border: red; }
/* Se establece el grosor y el color del borde, pero no
   su estilo, por lo que es "none" y el borde no se muestra */
div { border-bottom: 5px blue; }
```

Cuando los cuatro bordes no son idénticos pero sí muy parecidos, se puede utilizar la propiedad border para establecer de forma directa los atributos comunes de todos los bordes y posteriormente especificar para cada uno de los cuatro bordes sus propiedades particulares:

```
h1 {   border: solid #000;
       border-top-width: 6px;
       border-left-width: 8px;    }
```

Veamos un ejemplo con lo anterior: **Ej04.03e-BordesShortHand.html**

```
<!DOCTYPE html> <html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Bordes ShortHand </title>
   <style>
      h1 {  /* Si se indica con border estilos y colores en general los NO especificados se
             * ponen con 1px border: solid aqua;

            /* Estas 2 reglas ~ a los 3 inferiores*/
               border: solid white;
               border-width: 0px 1px 8px 1px;
            /* border-right:1px solid white;
               border-bottom: 8px solid white;
               border-left: 1px solid white; */      }

      p {   border: dashed Fuchsia;
            border-right-width: 10px;
            border-left-width: 5px;
      }
   </style>
</head>
<body>
<section>
      <h1> Ejemplos de Bordes</h1> <br /><br />
      <p> Esto es un borde realizado con propiedades <br />
            ShorHand que vemos en clase</p>
</section>
</body></html>
```

Ejercicio4
A partir del HTML original del Ejercicio3 (páginas 49 al 51):
Se pide (el resultado final se pone al final de esta página):

- Comentar la línea que incluye un fondo de textura para toda la página
- Eliminar el borde gris que muestran por defecto todos los elementos.
- El elemento #menu debe tener un borde inferior de 3 píxel y azul (#004C99).
- El elemento #noticias muestra un borde de 3 píxel y gris claro (#C5C5C5).
- El elemento #publicidad debe mostrar un borde discontinuo de 3 píxel y de color #CC6600.
- El lateral formado por el elemento #secundario muestra un borde de 3 px y color #CC6600.
- El elemento #pie debe mostrar un borde superior y otro inferior de 3 px y color #C5C5C5.

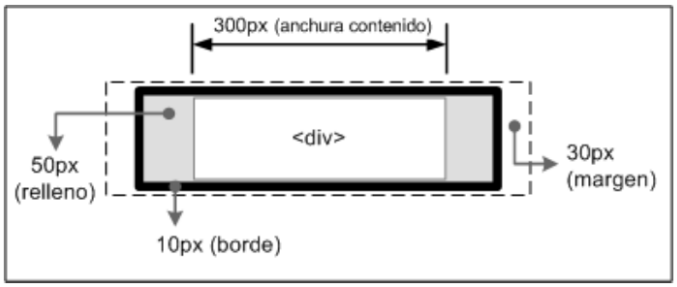
El resultado será el siguiente:

4.4 Margen, relleno, bordes y modelo de cajas

La anchura y altura de un elemento no solamente se calculan teniendo en cuenta sus propiedades width y height. El margen, el relleno y los bordes establecidos a un elemento determinan la anchura y altura final del elemento. En el siguiente ejemplo se muestran los estilos CSS de un elemento:

Otro ejemplo: **Ej04.04a-BordesCajas.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Bordes</title>
  <style type="text/css">
    div {
      width: 300px;
      padding-left:  50px;
      padding-right: 50px;
      margin-left:   30px;
      margin-right:  30px;
      border: 10px solid LightBlue;
    }
  </style>
</head>
<body>
    <section>
      <h1> Ejemplos de Bordes</h1>
      <div>
      <p>Lorem ipsum dolor ...</p>
      </div>
    </section>
</body>
</html>
```



La anchura total con la que se muestra el elemento no son los 300 píxel indicados en la propiedad width, sino que también se añaden todos sus márgenes, rellenos y bordes, como se indica en la imgan superior.

De esta forma, la anchura del elemento en pantalla sería igual a la suma de la anchura original, los márgenes, los bordes y los rellenos:

```
30px + 10px + 50px + 300px + 50px + 10px + 30px = 480 píxel
```

Así, la anchura/altura establecida con CSS siempre hace referencia a la anchura/altura del contenido. La anchura/altura total del elemento debe tener en cuenta además los valores del resto de partes que componen la caja del box model.

Por otra parte, la guerra de navegadores que se produjo en los años 90 provocó que cada fabricante (Microsoft y Netscape) añadiera sus propias extensiones y mejoras en sus productos. Posteriormente, aparecieron los estándares publicados por el W3C y los fabricantes se encontraron con el problema de la incompatibilidad entre sus implementaciones anteriores de HTML y CSS y las implementaciones que requerían los estándares.

La solución que adoptaron fue la de incluir en el navegador dos modos diferentes de funcionamiento: modo compatible con las páginas antiguas (denominado "modo quirks" y que se podría traducir como "modo raro") y modo compatible con los nuevos estándares (denominado "modo estándar"). El modo quirks es equivalente a la forma en la que se visualizaban las páginas en los navegadores Internet Explorer 4 y Netscape Navigator 4.

La diferencia más notable entre los dos modos es el tratamiento del "box model", lo que puede afectar gravemente al diseño de las páginas HTML. Los navegadores seleccionan automáticamente el modo en el que muestran las páginas en función del DOCTYPE definido por el documento.

En general, los siguientes tipos de DOCTYPE activan el modo quirks en los navegadores:

- No utilizar ningún DOCTYPE
- DOCTYPE anterior a HTML 4.0
  (`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">`)
- DOCTYPE de HTML 4.01 sin URL
  (`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">`)

En el caso concreto de Internet Explorer, también activan el modo quirks los modos XHTML 1.0 que incluyen la declaración de XML (por ejemplo <?xml version="1.0" encoding="UTF-8"?>) al principio de la página web:

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

Se pueden consultar todos los casos concretos que activan el modo quirks para cada navegador en la página http://hsivonen.iki.fi/doctype/

La versión 5.5 y anteriores de Internet Explorer y las versiones 6 y 7 en modo quirks siguen su propio modelo de cálculo de anchuras y alturas que es muy diferente al método definido por el estándar.

Para evitar este problema y crear diseños con el mismo aspecto en cualquier navegador, es necesario evitar el modo quirks de Internet Explorer. Por tanto, todas las páginas deben incluir la declaración apropiada de DOCTYPE.

4.4.1 Los modos de compatibilidad de Internet Explorer 8

El navegador Internet Explorer 8 introduce el concepto de "compatibilidad de la página" para asegurar que todas las páginas HTML se vean correctamente en cualquier versión de ese navegador. En realidad, esta nueva característica es una mejora del modo quirks explicado anteriormente.

Internet Explorer 8, a diferencia de sus versiones anteriores, soporta completamente el estándar CSS 2.1. Sin embargo, muchos sitios web se diseñaron para Internet Explorer 6 y 7, por lo que incluyen trucos, hacks y filtros que arreglan los errores y carencias de esas versiones del navegador.

Para evitar que las páginas diseñadas para navegadores anteriores se vean mal en esta nueva versión, Internet Explorer 8 incluye la opción de "compatibilidad de la página", que permite indicar la versión de Internet Explorer para la que la página ha sido diseñada.

De esta forma, si la página no se visualiza correctamente en Internet Explorer 8, se puede indicar al navegador que la muestre como si fuera Internet Explorer 6 o 7. En realidad, Internet Explorer 8 incluye seis modos de funcionamiento:

- **Modo IE5:** la página se muestra según el modo quirks de Internet Explorer 7, que es casi idéntico a como se veían las páginas en el navegador Internet Explorer 5.
- **Modo IE7:** la página se muestra en el modo estándar de Internet Explorer 7, sin importar si la página contiene o no la directiva <!DOCTYPE>.
- **Modo IE8:** los contenidos se muestran en el modo estándar de Internet Explorer 8, que es el más parecido al del resto de navegadores que soportan los estándares (Firefox, Opera, Safari y Google Chrome

- **Emular el modo IE7:** el navegador decide cómo mostrar los contenidos a partir de la directiva <!DOCTYPE> de la página. Si esa directiva es una de las que activan el modo estándar, la página se muestra en el modo estándar de Internet Explorer 7. En otro caso, se muestra en el modo quirks de Internet Explorer 5. Este modo es el más útil para la mayoría de sitios web.
- **Emular el modo IE8:** el navegador decide cómo mostrar los contenidos a partir de la directiva <!DOCTYPE> de la página. Si esa directiva es una de las que activan el modo estándar, la página se muestra en el modo estándar de Internet Explorer 8. En otro caso, se muestra en el modo quirks de Internet Explorer 5.
- **Modo límite ("edge mode"):** indica a Internet Explorer que los contenidos se deben mostrar en el modo de compatibilidad más avanzado disponible. Actualmente, este modo es equivalente al modo IE8. Si las futuras versiones Internet Explorer 9 y 10 incluyeran mejor compatibilidad, las páginas se visualizarían en ese modo avanzado de compatibilidad.

El modo de compatibilidad de la página se indica mediante una nueva etiqueta <meta> con la propiedad X-UA-Compatible y cuyo valor es el que utiliza Internet Explorer 8 para determinar el modo que se utiliza:

```
<!-- Modo IE5 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=5" />
  ...
</head>

<!-- Modo IE7 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=7" />
  ...
</head>

<!-- Modo IE8 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=8" />
  ...
</head>

<!-- Emular el modo IE7 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
  ...
</head>

<!-- Emular el modo IE8 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8" />
  ...
</head>

<!-- Modo límite -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  ...
</head>
```

No obstante, esta opción de compatibilidad de la página debe entenderse como una solución temporal que evita que los sitios web se vean mal en Internet Explorer 8. La única solución correcta a largo plazo consiste en actualizar las páginas para que sus diseños sigan los estándares web.

4.5 Fondos

El último elemento que forma el box model es el fondo de la caja del elemento. El fondo puede ser un color simple o una imagen. El fondo solamente se visualiza en el área ocupada por el contenido y su relleno, ya que el color de los bordes se controla directamente desde los bordes y las zonas de los márgenes siempre son transparentes.

Para establecer un color o imagen de fondo en la página entera, se debe establecer un fondo al elemento <body>. Si se establece un fondo a la página, como el valor inicial del fondo de los elementos es transparente, todos los elementos de la página se visualizan con el mismo fondo a menos que algún elemento especifique su propio fondo.

CSS define cinco propiedades para establecer el fondo de cada elemento (background-color, background-image, background-repeat, background-attachment, background-position) y otra propiedad de tipo "shorthand" (background).

La propiedad background-color permite mostrar un color de fondo sólido en la caja de un elemento. Esta propiedad no permite crear degradados ni ningún otro efecto avanzado.

Propiedad	**background-color**
Valores	<color> | transparent | inherit
Se aplica a	Todos los elementos
Valor inicial	transparent
Descripción	Establece un color de fondo para los elementos

El siguiente ejemplo muestra una página web con un color gris claro de fondo:

```
body {
  background-color: #F5F5F5;
}
```

Para crear efectos gráficos avanzados, es necesario utilizar la propiedad background-image, que permite mostrar una imagen como fondo de la caja de cualquier elemento:

CSS permite establecer de forma simultánea un color y una imagen de fondo. En este caso, la imagen se muestra delante del color, por lo que solamente si la imagen contiene zonas transparentes es posible ver el color de fondo.

El siguiente ejemplo muestra una imagen como fondo de toda la página:

```
body { background-image: url("imagenes/fondo.png") }
```

Las imágenes de fondo se indican a través de su URL, que puede ser absoluta o relativa. Suele ser recomendable crear una carpeta de imágenes que se encuentre en el mismo directorio que los archivos CSS y que almacene todas las imágenes utilizadas en el diseño de las páginas.

Así, las imágenes correspondientes al diseño de la página se mantienen separadas del resto de imágenes del sitio y el código CSS es más sencillo (por utilizar URL relativas) y más fácil de mantener (por no tener que actualizar URL absolutas en caso de que se cambie la estructura del sitio web).

Por otra parte, suele ser habitual indicar un color de fondo siempre que se muestra una imagen de fondo. En caso de que la imagen no se pueda mostrar o contenga errores, el navegador mostrará el color indicado (que debería ser, en lo posible, similar a la imagen) y la página no parecerá que contiene errores.

Si la imagen que se quiere mostrar es demasiado grande para el fondo del elemento, solamente se muestra la parte de imagen comprendida en el tamaño del elemento. Si la imagen es más pequeña que el elemento, CSS la repite horizontal y verticalmente hasta llenar el fondo del elemento.

Este comportamiento es útil para establecer un fondo complejo a una página web entera. El siguiente ejemplo utiliza una imagen muy pequeña para establecer un fondo complejo a toda una página, además de añadir otro fondo para el DIV.

En primer lugar nos descargamos el fondo del body que será una textura:
http://www.bancodeimagenesgratis.com/2011/09/texturas-para-diseno-grafico-blogs-y.html
Para el DIV usaremos un fondo como este:
http://www.bancodeimagenesgratis.com/2012/11/delfines-nadando-en-el-mar-azul.html
Otro ejemplo: **Ej04.05a-FondoTextura.html**

```
<!DOCTYPE html> <html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Fondos</title>
   <style>
      body {   background-image: url("img/Textura4.png");   }
      #caja1   {      position: absolute;
                      top: 100px;    left: 200px;
                      width: 800px;  height: 600px;
                      border: 10px solid orange;
                      background-image: url("img/Delfines.jpg");
                      /* Le ponemos una transparencia del 70% */
                      opacity: 0.7;         }

      #caja2   {      position: absolute;
                      top: 100px;    left: 200px;
                      width: 800px;  height: 600px;
                      color: yellow;        }

      p        {      padding: 30px;
                      font-size: 30px;             }
      #p2      {      text-align: center;   font-size: 50px;      }
   </style>
</head>
<body>
      <section>
         <div id="caja1"></div>
         <div id="caja2">
            <p> Esto es una caja con <br />  un Fondo de Delfines.</p>
            <br> <p id="p2">OLE MI DELFIN!!</p>
         </div>
      </section>
</body> </html>
```

Con una imagen muy pequeña (y que por tanto, se puede descargar en muy poco tiempo) se consigue cubrir completamente el fondo de la página, con lo que se consigue un gran ahorro de ancho de banda.

En ocasiones, no es conveniente que la imagen de fondo se repita horizontal y verticalmente. Para ello, CSS introduce la propiedad background-repeat que permite controlar la forma de repetición de las imágenes de fondo.

Propiedad	**background-repeat**
Valores	repeat | repeat-x | repeat-y | no-repeat | inherit
Se aplica a	Todos los elementos
Valor inicial	repeat
Descripción	Controla la forma en la que se repiten las imágenes de fondo

El valor repeat indica que la imagen se debe repetir en todas direcciones y por tanto, es el comportamiento por defecto. El valor no-repeat muestra una sola vez la imagen y no se repite en ninguna dirección. El valor repeat-x repite la imagen sólo horizontalmente y el valor repeat-y repite la imagen solamente de forma vertical.

Por ejemplo, si queremos implementar una cabecera, podríamos usar la textura descargada del modo siguiente:
Otro ejemplo: **Ej04.05b-FondoCabecera.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Fondos</title>
  <style type="text/css">
    body {
      background-color: LightCyan;
    }

    nav {
      background-image:url(img/Textura4.png);
      background-repeat: repeat-x;
      width: 100%;
      height: 25px;
      text-align: center;
      color: white;
    }
  </style>
</head>
<body>
    <nav> Portada | Seccion1 | Sección 2 </nav>
    <section>
      <h1> Ejemplos de Fondos</h1>
      <div>
   <p>Lorem ipsum dolor ...</p>
      </div>
    </section>
</body>
</html>
```

Además de seleccionar el tipo de repetición de las imágenes de fondo, CSS permite controlar la posición de la imagen dentro del fondo del elemento mediante la propiedad background-position.

Propiedad	**background-position**
Valores	((<porcentaje> | <medida> | left | center | right) (<porcentaje> | <medida> | top | center | bottom)?) | ((left | center | right) || (top | center | bottom)) | inherit
Se aplica a	Todos los elementos
Valor inicial	0% 0%
Descripción	Controla la posición en la que se muestra la imagen en el fondo del elemento

La propiedad background-position permite indicar la distancia que se desplaza la imagen de fondo respecto de su posición original situada en la esquina superior izquierda.

Si se indican dos porcentajes o dos medidas, el primero indica el desplazamiento horizontal y el segundo el desplazamiento vertical respecto del origen (situado en la esquina superior izquierda). Si solamente se indica un porcentaje o una medida, se considera que es el desplazamiento horizontal y al desplazamiento vertical se le asigna automáticamente el valor de 50%.

Cuando se utilizan porcentajes, su interpretación no es intuitiva. Si el valor de la propiedad background-position se indica mediante dos porcentajes x% y%, el navegador coloca el punto (x%, y%) de la imagen de fondo en el punto (x%, y%) del elemento.

Las palabras clave permitidas son equivalentes a algunos porcentajes significativos: top = 0%, left = 0%, center = 50%, bottom = 100%, right = 100%.

CSS permite mezclar porcentajes y palabras clave, como por ejemplo 50% 2cm, center 2cm, center 10%.

Si se utilizan solamente palabras clave, el orden es indiferente y por tanto, es equivalente indicar top left y left top.

El siguiente ejemplo muestra una misma imagen de fondo posicionada de tres formas diferentes. En primer lugar guardaremos la siguiente imagen:
http://www.gettyicons.com/free-icons/112/must-have/png/256/help_256.png
Lo enviamos a la carpeta img y le cambiamos el nombre a help.png.

Otro ejemplo: **Ej04.05c-FondoPosition.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Fondo Position</title>
   <style>
      #contenedor {
         width: 1300px;
      }

      #caja1 {        background-image: url("img/help.png");
                      background-repeat: no-repeat;
                      background-position: bottom left;

                      width: 400px;                 height: 600px;
                      border: 5px solid blue;       float: left;
      }

      #caja2 {               background-image: url("img/help.png");
                      background-repeat: no-repeat;
                      background-position: center;

                      width: 400px;                 height: 600px;
                      border: 5px solid blue;       float: left;
                      margin: 0px 35px;
      }

      #caja3 {               background-image: url("img/help.png");
                      background-repeat: no-repeat;
                      background-position: top right;

                      width: 400px;                 height: 600px;
                      border: 5px solid blue;       float: right;
      }
   </style>
</head>

<body>
      <section>
         <div id="contenedor">
            <div id="caja1"><h1>Abajo Izquierda</h1></div>
            <div id="caja2"><h1>Centrado</h1></div>
            <div id="caja3"><h1>Arriba Derecha</h1></div>
         </div>
      </section>
</body>
</html>
```

Opcionalmente, se puede indicar que el fondo permanezca fijo cuando la ventana del navegador se desplaza mediante las barras de scroll. Se trata de un comportamiento que en general no es deseable y que algunos navegadores no soportan correctamente. La propiedad que controla este comportamiento es background-attachment.

Propiedad	**background-attachment**
Valores	scroll | fixed | inherit
Se aplica a	Todos los elementos
Valor inicial	scroll
Descripción	Controla la forma en la que se visualiza la imagen de fondo: permanece fija cuando se hace scroll en la ventana del navegador o se desplaza junto con la ventana

Para hacer que una imagen de fondo se muestre fija al desplazar la ventana del navegador, se debe añadir la propiedad background-attachment: fixed. **Ej04.05d-FondoAttachment.html**

```
...
    body {
         background-image: url("img/Delfines.jpg");
         background-attachment: scroll;
      }
...
    <section>
      <div id="contenedor"> </div>
    </section>
</body> </html>
```

Por último, CSS define una propiedad de tipo "shorthand" para indicar todas las propiedades de los colores e imágenes de fondo de forma directa. La propiedad se denomina background y es la que generalmente se utiliza para establecer las propiedades del fondo de los elementos.

Propiedad	**background**
Valores	(<background-color> || <background-image> || <background-repeat> || <background-attachment> || <background-position>) | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece todas las propiedades del fondo de un elemento

El orden en el que se indican las propiedades es indiferente, aunque en general se sigue el formato indicado de color, url de imagen, repetición y posición. El siguiente ejemplo muestra la ventaja de utilizar la propiedad background:

```
/* Color e imagen de fondo de la página mediante una propiedad shorthand */
body { background: #222d2d url(./graphics/colorstrip.gif) repeat-x 0 0; }
/* La propiedad shorthand anterior es equivalente a las siguientes propiedades */
body {        background-color: #222d2d;
              background-image: url("./graphics/colorstrip.gif");
              background-repeat: repeat-x;
              background-position: 0 0;
}
```

La propiedad background permite asignar todos o sólo algunos de todos los valores que se pueden definir para los fondos de los elementos:

```
background: url("./graphics/wide/bg-content-secondary.gif") repeat-y;
background: url("./graphics/wide/footer-content-secondary.gif") no-repeat bottom
left;
background: transparent url("./graphics/navigation.gif") no-repeat 0 -27px;
background: none;
background: #293838 url("./graphics/icons/icon-permalink-big.gif") no-repeat
center left;
```

Ejercicio5

A partir del HTML y del CSS del Ejercicio 4 RESUELTO (ver páginas 58 y 59) se pide:

- Los elementos #noticias y #pie tiene un color de fondo gris claro (#D8D8D8).
- El elemento #publicidad muestra un color de fondo amarillo claro (#FFF6CD).
- Los elementos <h2> del lateral #secundario muestran un color de fondo #DB905C y un pequeño padding de .2em.
- El fondo del elemento #menu se construye mediante una pequeña imagen llamada fondo_menu.gif.
- El logotipo del sitio se muestra mediante una imagen de fondo del elemento <h1> contenido en el elemento #logo (la imagen se llama logo.gif), que no se repite, situado a -10px del Top y -10px de la izquierda.
- Cambiar #logo para que tenga una sangría de texto (text-indent) de 200px y altura 35px.
- Por último, eliminar en el HTML el texto Logotipo y dejar un espacio en blanco en el span del h1 dentro de #logo.

Las imágenes a descargar son:
http://librosweb.es/ejercicios/css/comun/imagenes/fondo_menu.gif
http://librosweb.es/ejercicios/css/comun/imagenes/logo.gif

Resultado:

5 POSICIONAMIENTO Y VISUALIZACIÓN

Cuando los navegadores descargan el contenido HTML y CSS de las páginas web, aplican un procesamiento muy complejo antes de mostrar las páginas en la pantalla del usuario.

Para cumplir con el modelo de cajas presentado en el capítulo anterior, los navegadores crean una caja para representar a cada elemento de la página HTML. Los factores que se tienen en cuenta para generar cada caja son:

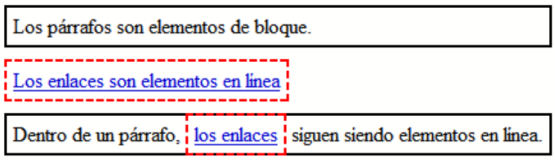
- Las propiedades width y height de la caja (si están establecidas).
- El tipo de cada elemento HTML (elemento de bloque o elemento en línea).
- Posicionamiento de la caja (normal, relativo, absoluto, fijo o flotante).
- Las relaciones entre elementos (dónde se encuentra cada elemento, elementos descendientes, etc.)
- Otro tipo de información, como por ejemplo el tamaño de las imágenes y el tamaño de la ventana del navegador.

5.1 Tipos de elementos

El estándar HTML clasifica a todos sus elementos en dos grandes grupos: elementos en línea y elementos de bloque.

Los elementos de bloque ("block elements" en inglés) siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea. Por su parte, los elementos en línea ("inline elements" en inglés) no empiezan necesariamente en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

Debido a este comportamiento, el tipo de un elemento influye de forma decisiva en la caja que el navegador crea para mostrarlo. La siguiente imagen muestra las cajas que crea el navegador para representar los diferentes elementos que forman una página HTML:



El primer elemento de la página anterior es un párrafo. Los párrafos son elementos de bloque y por ese motivo su caja empieza en una nueva línea y llega hasta el final de esa misma línea. Aunque los contenidos de texto del párrafo no son suficientes para ocupar toda la línea, el navegador reserva todo el espacio disponible en la primera línea.

El segundo elemento de la página es un enlace. Los enlaces son elementos en línea, por lo que su caja sólo ocupa el espacio necesario para mostrar sus contenidos. Si después de este elemento se incluye otro elemento en línea (por ejemplo otro enlace o una imagen) el navegador mostraría los dos elementos en la misma línea, ya que existe espacio suficiente.

Por último, el tercer elemento de la página es un párrafo que se comporta de la misma forma que el primer párrafo. En su interior, se encuentra un enlace que también se comporta de la misma forma que el enlace anterior. Así, el segundo párrafo ocupa toda una línea y el segundo enlace sólo ocupa el espacio necesario para mostrar sus contenidos.

Por sus características, los elementos de bloque no pueden insertarse dentro de elementos en línea y tan sólo pueden aparecer dentro de otros elementos de bloque. En cambio, un elemento en línea puede aparecer tanto dentro de un elemento de bloque como dentro de otro elemento en línea.

Los elementos en línea definidos por HTML son: `a, abbr, acronym, b, basefont, bdo, big, br, cite, code, dfn, em, font, i, img, input, kbd, label, q, s, samp, select, small, span, strike, strong, sub, sup, textarea, tt, u, var.`

Los elementos de bloque definidos por HTML son: `address, blockquote, center, dir, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, isindex, menu, noframes, noscript, ol, p, pre, table, ul.`

Los siguientes elementos también se considera que son de bloque: `dd, dt, frameset, li, tbody, td, tfoot, th, thead, tr.`

Los siguientes elementos pueden ser en línea y de bloque según las circunstancias: `button, del, iframe, ins, map, object, script.`

5.2 Posicionamiento

Los navegadores crean y posicionan de forma automática todas las cajas que forman cada página HTML. No obstante, CSS permite al diseñador modificar la posición en la que se muestra cada caja.

Utilizando las propiedades que proporciona CSS para alterar la posición de las cajas es posible realizar efectos muy avanzados y diseñar estructuras de páginas que de otra forma no serían posibles.

El estándar de CSS define cinco modelos diferentes para posicionar una caja:

- Posicionamiento normal o estático: se trata del posicionamiento que utilizan los navegadores si no se indica lo contrario.
- Posicionamiento relativo: variante del posicionamiento normal que consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.
- Posicionamiento absoluto: la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.
- Posicionamiento fijo: variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.
- Posicionamiento flotante: se trata del modelo más especial de posicionamiento, ya que desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

El posicionamiento de una caja se establece mediante la propiedad position:

Propiedad	**position**
Valores	static | relative | absolute | fixed | inherit
Se aplica a	Todos los elementos
Valor inicial	static
Descripción	Selecciona el posicionamiento con el que se mostrará el elemento

El significado de cada uno de los posibles valores de la propiedad position es el siguiente:

- static: corresponde al posicionamiento normal o estático. Si se utiliza este valor, se ignoran los valores de las propiedades top, right, bottom y left.
- relative: corresponde al posicionamiento relativo. El desplazamiento de la caja se controla con las propiedades top, right, bottom y left.
  Se considera el elemento anterior excepto si este es absoluto.
- absolute: corresponde al posicionamiento absoluto. El desplazamiento de la caja también se controla con las propiedades top, right, bottom y left, pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor.
- fixed: corresponde al posicionamiento fijo. El desplazamiento se establece de la misma forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

La propiedad position no permite controlar el posicionamiento flotante, que se establece con otra propiedad llamada float y que se explica más adelante. Además, la propiedad position sólo indica cómo se posiciona una caja, pero no la desplaza.

Normalmente, cuando se posiciona una caja también es necesario desplazarla respecto de su posición original o respecto de otro origen de coordenadas. CSS define cuatro propiedades llamadas top, right, bottom y left para controlar el desplazamiento de las cajas posicionadas:

Propiedades	**top, right, bottom, left**
Valores	<medida> | <porcentaje> | auto | inherit
Se aplica a	Todos los elementos posicionados
Valor inicial	auto
Descripción	Indican el desplazamiento horizontal y vertical del elemento respecto de su posición original

En el caso del posicionamiento relativo, cada una de estas propiedades indica el desplazamiento del elemento desde la posición original de su borde superior/derecho/inferior/izquierdo. Si el posicionamiento es absoluto, las propiedades indican el desplazamiento del elemento respecto del borde superior/derecho/inferior/izquierdo de su primer elemento padre posicionado.

En cualquiera de los dos casos, si el desplazamiento se indica en forma de porcentaje, se refiere al porcentaje sobre la anchura (propiedades right y left) o altura (propiedades top y bottom) del elemento. Veamos varios ejemplos. **Ej05.02a-Posicionamiento.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Posicionamiento</title>
  <style type="text/css">
    #cont {    border: 20px solid red;
               width: 400px; height: 300px;
               margin: 20px;  }
   #caja {     border: 5px double blue;
               width: 200px;  height: 200px;
               /* Jugamos con la posición: relative/absolute */
               position: relative;
               top: 50px;     left: 50px;
    }
  </style>
</head>
<body>
    <section>
      <div id="cont">
         <div id="caja"> <p>Lorem ipsum dolor.</p> </div>
      </div>
    </section>
</body> </html>
```

Veamos otros dos ejemplos mas aplicados a tablas:

Ej05.02b-PosAbsoluto.html

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Posicionamiento Absoluto</title>
  <style type="text/css">
    #absoluto {   position:absolute;
                  top:18px;   left:40px;     }

    table { width: 100%;      border-collapse:collapse;
            border: 2px solid blue;  }
    td {    padding:0px;      /* Equivale a cellpadding="0px" para el Table*/
            /* Equivale a cellspacing="0px", junto al border-collapse: collapse*/
            margin:0px;       border: 1px solid red;               }
  </style>
</head>
<body>
    <section>
      <!-- El DIV toma como referencia su contenedor (section); lo demás NO IMPORTA -->
      <div id="absoluto">Posición absoluta</div>
      <table >       <!-- El border-collapse PEGA las celdas unas a otras -->
   <tr>
     <td> </td>
     <td> </td>
   </tr>
   <tr>
     <td> </td>
     <td> </td>
   </tr>
      </table>
    </section>
</body>
</html>
```

Ej05.02c-PosRelativo.html

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Posicionamiento Relativo</title>
  <style type="text/css">
    #relativo {   position:relative;
                  top:20px;   left:30px;     }
    #relativo2 {  position:relative;
                  top:-20px;  left:30px;     }
    table {       width: 100%;       border-collapse:collapse;
                  border: 2px solid blue;    }
    td {             padding:0px;    margin:0px;
                  border: 1px solid red;     }
  </style>
</head>
<body>
    <section>
      <!-- Se coloca a 20px del top y 30px del left respecto al inicio de la pag -->
      <span id="relativo">Posición Relativa</span>
      <table >
      <tr>
         <td> </td>
         <td> </td>
      </tr>
      <tr>
         <td> </td>
         <td> </td>
      </tr>
      </table>
      <!-- Respecto al FINAL DE LA TABLA se pone 20px HACIA ARRIBA y 30px A LA IZQ -->
      <div id="relativo2">Otra posición relativa</div>
    </section>
</body>
</html>
```

5.3 Posicionamiento normal

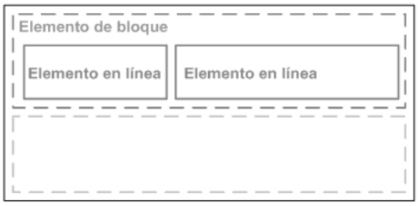
El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, sólo se tiene en cuenta si el elemento es de bloque o en línea, sus propiedades width y height y su contenido.

Los elementos de bloque forman lo que CSS denomina "contextos de formato de bloque". En este tipo de contextos, las cajas se muestran una debajo de otra comenzando desde el principio del elemento contenedor. La distancia entre las cajas se controla mediante los márgenes verticales.

Si un elemento se encuentra dentro de otro, el elemento padre se llama "elemento contenedor" y determina tanto la posición como el tamaño de todas sus cajas interiores.

Si un elemento no se encuentra dentro de un elemento contenedor, entonces su elemento contenedor es el elemento <body> de la página. Normalmente, la anchura de los elementos de bloque está limitada a la anchura de su elemento contenedor, aunque en algunos casos sus contenidos pueden desbordar el espacio disponible.

Los elementos en línea forman los "contextos de formato en línea". En este tipo de contextos, las cajas se muestran una detrás de otra de forma horizontal comenzando desde la posición más a la izquierda de su elemento contenedor. La distancia entre las cajas se controla mediante los márgenes laterales.



Si las cajas en línea ocupan más espacio del disponible en su propia línea, el resto de cajas se muestran en las líneas inferiores. Si las cajas en línea ocupan un espacio menor que su propia línea, se puede controlar la distribución de las cajas mediante la propiedad text-align para centrarlas, alinearlas a la derecha o justificarlas.

Un ejemplo. **Ej05.03a-PosicStatic.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Posicionamiento Estático</title>
  <style type="text/css">
    #caja1 {   position: static;
               background-color: #ff9; padding: 10px; width: 300px;       }
    #caja2 {   position: static;
               background-color: #f9f; padding: 10px; width: 500px;       }

    #caja3 {   background-color: #9ff; padding: 10px; width: 400px;       }
  </style>
</head>
<body>
    <section>
      <div id="caja1">Esto es una capa con posicionamiento estático</div>
      <div id="caja2">posicionamiento static, predeterminado.</div>
      <h1>CSS</h1>
      <div id="caja3">Posicionamiento static, aunque en este caso no se indicó
      el atributo position static, pues no hace falta.</div>
    </section>
</body>
</html>
```

5.4 Posicionamiento relativo

El estándar CSS considera que el posicionamiento relativo es un caso particular del posicionamiento normal, aunque en la práctica presenta muchas diferencias.

El posicionamiento relativo desplaza una caja respecto de su posición original establecida mediante el posicionamiento normal. El desplazamiento de la caja se controla con las propiedades top, right, bottom y left.

El valor de la propiedad top se interpreta como el desplazamiento entre el borde superior de la caja en su posición final y el borde superior de la misma caja en su posición original.

De la misma forma, el valor de las propiedades left, right y bottom indica respectivamente el desplazamiento entre el borde izquierdo/derecho/inferior de la caja en su posición final y el borde izquierdo/derecho/inferior de la caja original.

Por tanto, la propiedad top se emplea para mover las cajas de forma descendente, la propiedad bottom mueve las cajas de forma ascendente, la propiedad left se utiliza para desplazar las cajas hacia la derecha y la propiedad right mueve las cajas hacia la izquierda. Este comportamiento parece poco intuitivo y es causa de errores cuando se empiezan a diseñar páginas con CSS. Si se utilizan valores negativos en las propiedades top, right, bottom y left, su efecto es justamente el inverso.

El desplazamiento relativo de una caja no afecta al resto de cajas adyacentes, que se muestran en la misma posición que si la caja desplazada no se hubiera movido de su posición original.

En la imagen anterior, la caja 2 se ha desplazado lateralmente hacia la derecha y verticalmente de forma descendente. Como el resto de cajas de la página no modifican su posición, se producen solapamientos entre los contenidos de las cajas.

Las cajas desplazadas de forma relativa no modifican su tamaño, por lo que los valores de las propiedades left y right siempre cumplen que left = -right.

Si tanto left como right tienen un valor de auto (que es su valor por defecto) la caja no se mueve de su posición original. Si sólo el valor de left es auto, su valor real es -right. Igualmente, si sólo el valor de right es auto, su valor real es -left.

Si tanto left como right tienen valores distintos de auto, uno de los dos valores se tiene que ignorar porque son mutuamente excluyentes. Para determinar la propiedad que se tiene en cuenta, se considera el valor de la propiedad direction.

La propiedad direction permite establecer la dirección del texto de un contenido. Si el valor de direction es ltr, el texto se muestra de izquierda a derecha, que es el método de escritura habitual en la mayoría de países. Si el valor de direction es rtl, el método de escritura es de derecha a izquierda, como el utilizado por los idiomas árabe y hebreo.

Si el valor de direction es ltr, y las propiedades left y right tienen valores distintos de auto, se ignora la propiedad right y sólo se tiene en cuenta el valor de la propiedad left. De la misma forma, si el valor de direction es rtl, se ignora el valor de left y sólo se tiene en cuenta el valor de right.

Un ejemplo. **Ej05.04a-PosicRelativo.html**

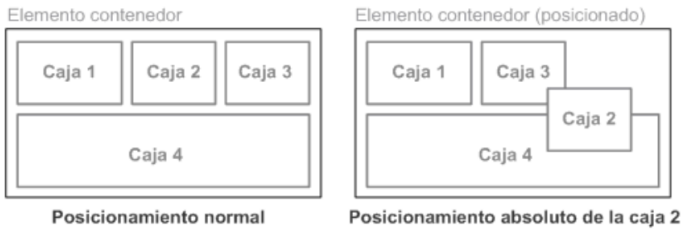
```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Posicionamiento Relativo</title>
  <style type="text/css">
    img {
      width: 20%;
    }

    img.desplazada {
      position: relative;
      top: 8em;
    }

  </style>
</head>
<body>
    <section>
      <img class="desplazada" src="img/roma.jpg" alt="Imagen genérica" />
      <img src="img/roma.jpg" alt="Imagen genérica" />
      <img src="img/roma.jpg" alt="Imagen genérica" />
    </section>
</body>
</html>
```

El resto de imágenes no varían su posición y por tanto no ocupan el hueco dejado por la primera imagen, ya que el posicionamiento relativo no influye en el resto de elementos de la página. El principal problema de posicionar elementos de forma relativa es que se pueden producir solapamientos con otros elementos de la página.

5.5 Posicionamiento absoluto

El posicionamiento absoluto se emplea para establecer de forma exacta la posición en la que se muestra la caja de un elemento. La nueva posición de la caja se indica mediante las propiedades top, right, bottom y left. La interpretación de los valores de estas propiedades es mucho más compleja que en el posicionamiento relativo, ya que en este caso dependen del posicionamiento del elemento contenedor.

Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página se ven afectados y modifican su posición. Al igual que en el posicionamiento relativo, cuando se posiciona de forma absoluta una caja es probable que se produzcan solapamientos con otras cajas.



Posicionamiento normal
Posicionamiento absoluto de la caja 2

En el siguiente ejemplo, se posiciona de forma absoluta la caja 2:

La caja 2 está posicionada de forma absoluta, lo que provoca que el resto de elementos de la página modifiquen su posición. En concreto, la caja 3 deja su lugar original y pasa a ocupar el hueco dejado por la caja 2.

El estándar de CSS 2.1 indica que las cajas posicionadas de forma absoluta "salen del flujo normal de la página", lo que provoca que el resto de elementos de la página se muevan y en ocasiones, ocupen la posición original en la que se encontraba la caja.

Por otra parte, el desplazamiento de una caja posicionada de forma absoluta se controla mediante las propiedades top, right, bottom y left. A diferencia del posicionamiento relativo, la interpretación de los valores de estas propiedades depende del elemento contenedor de la caja posicionada.

Determinar la referencia utilizada para interpretar los valores de top, right, bottom y left de una caja posicionada de forma absoluta es un proceso complejo que se compone de los siguientes pasos:

- Se buscan todos los elementos contenedores de la caja hasta llegar al elemento <body> de la página.
- Se recorren todos los elementos contenedores empezando por el más cercano a la caja y llegando hasta el <body>
- El primer elemento contenedor que esté posicionado de cualquier forma diferente a position: static se convierte en la referencia que determina la posición de la caja posicionada de forma absoluta.
- Si ningún elemento contenedor está posicionado, la referencia es la ventana del navegador, que no debe confundirse con el elemento <body> de la página.

Una vez determinada la referencia del posicionamiento absoluto, la interpretación de los valores de las propiedades top, right, bottom y left se realiza como sigue:

- El valor de la propiedad top indica el desplazamiento desde el borde superior de la caja hasta el borde superior del elemento contenedor que se utiliza como referencia.
- El valor de la propiedad right indica el desplazamiento desde el borde derecho de la caja hasta el borde derecho del elemento contenedor que se utiliza como referencia.
- El valor de la propiedad bottom indica el desplazamiento desde el borde inferior de la caja hasta el borde inferior del elemento contenedor que se utiliza como referencia.
- El valor de la propiedad left indica el desplazamiento desde el borde izquierdo de la caja hasta el borde izquierdo del elemento contenedor que se utiliza como referencia.

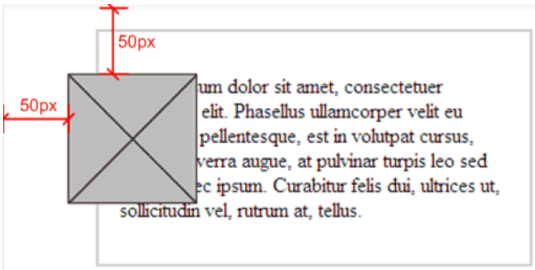
Primero la posición original. **Ej05.05a-PosicOriginal.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Posicionamiento Relativo</title>
  <style type="text/css">
    img {   width: 200px;     }

    div {   border: 2px solid #CCC;
      padding: 1em;
      margin: 1em 0 1em 4em;
      width: 300px;    }
 </style>
</head>
<body>
    <section>
      <div>
      <img src="img/roma.jpg" alt="Imagen genérica" />
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. </p>
      </div>
    </section>
</body>
</html>
```

Y ahora el cambio: **Ej05.05b-PosicAbsoluto.html**

```
...
  <style type="text/css">
...
      /* Añadimos la siguiente regla*/
    div img {  position: absolute;
               top:  50px;    left: 50px;    }
 </style>
</head>
<body>
    <section>
  ...
</html>
```

La imagen posicionada de forma absoluta no toma como referencia su elemento contenedor , sino la ventana del navegador, tal y como demuestra la siguiente imagen:



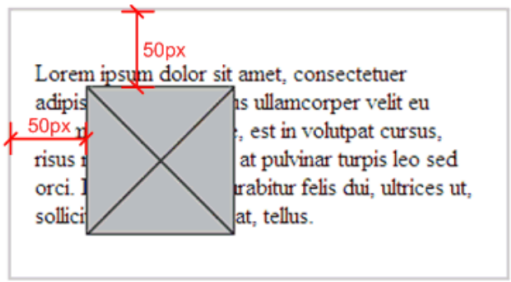
Para posicionar la imagen de forma absoluta, el navegador realiza los siguientes pasos:

1. Obtiene la lista de elementos contenedores de la imagen: y <body>.
2. Recorre la lista de elementos contenedores desde el más cercano a la imagen (el) hasta terminar en el <body> buscando el primer elemento contenedor que esté posicionado.
3. El posicionamiento de todos los elementos contenedores es el normal o estático, ya que ni siquiera tienen establecida la propiedad position
4. Como ningún elemento contenedor está posicionado, la referencia es la ventana del navegador (ojo, no el body).
5. A partir de esa referencia, la caja de la imagen se desplaza 50px hacia la derecha (left: 50px) y otros 50px de forma descendente (top: 50px) respecto a la esquina superior izquierda.

Como la imagen se posiciona de forma absoluta, el resto de elementos de la página se mueven para ocupar el lugar libre dejado por la imagen. Por este motivo, el párrafo sube hasta el principio del y se produce un solapamiento con la imagen posicionada que impide ver parte de los contenidos del párrafo.

A continuación, se modifica el ejemplo anterior posicionando de forma relativa el elemento que contiene la imagen y el párrafo. La única propiedad añadida al es position: relative por lo que el elemento contenedor se posiciona pero no se desplaza respecto de su posición original:
Otro cambio: **Ej05.05c-PosicRelative.html**

```
...
  <style type="text/css">
...
      /* Dejamos estos estilos*/
    img {   width: 200px;     }

    div {   border: 2px solid #CCC;
            padding: 1em;     margin: 1em 0 1em 4em;
            width: 300px;
            position: relative;       }

    div img {  position: absolute;
            top: 50px; left: 50px;    }
 </style>
</head>
<body>
    <section>
  ...
</html>
```



En este caso, como el elemento contenedor de la imagen está posicionado, se convierte en la referencia para el posicionamiento absoluto. El resultado es que la posición de la imagen es muy diferente a la del ejemplo anterior como aparece en la imagen superior.

Por tanto, si se quiere posicionar un elemento de forma absoluta respecto de su elemento contenedor, es imprescindible posicionar este último. Para ello, sólo es necesario añadir la propiedad position: relative, por lo que no es obligatorio desplazar el elemento contenedor respecto de su posición original.

5.6 Posicionamiento fijo

El estándar CSS considera que el posicionamiento fijo es un caso particular del posicionamiento absoluto, ya que sólo se diferencian en el comportamiento de las cajas posicionadas.

Cuando una caja se posiciona de forma fija, la forma de obtener el origen de coordenadas para interpretar su desplazamiento es idéntica al posicionamiento absoluto. De hecho, si el usuario no mueve la página HTML en la ventana del navegador, no existe ninguna diferencia entre estos dos modelos de posicionamiento.

La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador. El posicionamiento fijo hace que las cajas no modifiquen su posición ni aunque el usuario suba o baje la página en la ventana de su navegador.

Si la página se visualiza en un medio paginado (por ejemplo en una impresora) las cajas posicionadas de forma fija se repiten en todas las páginas. Esta característica puede ser útil para crear encabezados o pies de página en páginas HTML preparadas para imprimir.

El posicionamiento fijo apenas se ha utilizado en el diseño de páginas web hasta hace poco tiempo porque el navegador Internet Explorer 6 y las versiones anteriores no lo soportan.

Sigamos con el ejemplo anterior: **Ej05.06a-PosicFijo.html**

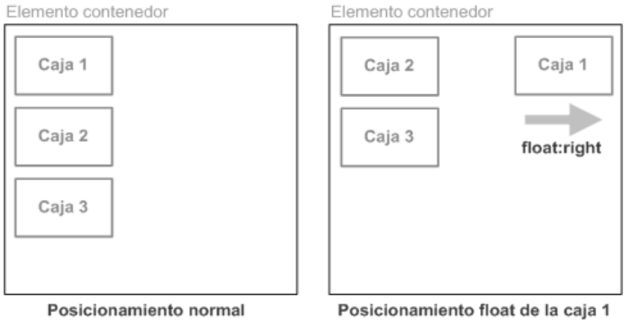
```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Posicionamiento Relativo</title>
  <style type="text/css">
    img {   width: 200px;     }

    div {
      border: 2px solid #CCC;
      padding: 1em;
      margin: 1em 0 1em 4em;
      width: 300px;
      position: relative;
    }

    div img {
      position: fixed;
      top: 50px;
      left: 50px;
    }

  </style>
</head>
<body>
    <section>
      <div>
      <img src="img/roma.jpg" alt="Imagen genérica" />
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. </p>
      </div>
      <p>  </p><p>  </p><p>  </p><p>  </p><p>  </p>
      <p>  </p><p>  </p><p>  </p><p>  </p><p>  </p>
      <p>  </p><p>  </p><p>  </p><p>  </p><p>  </p>
    </section>
</body>
</html>
```

Obsérvese que, si se amplía el zoom, y se baja con la barra de desplazamiento, la imagen se mantiene en la misma posición.

5.7 Posicionamiento flotante

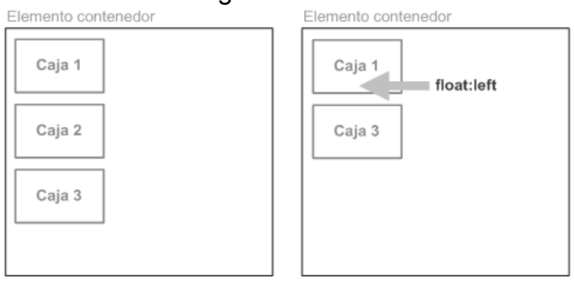
El posicionamiento flotante es el más difícil de comprender pero al mismo tiempo es el más utilizado. La mayoría de estructuras de las páginas web complejas están diseñadas con el posicionamiento flotante, como se verá más adelante.

Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una caja flotante, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.
La imagen derecha muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1.



Posicionamiento normal
Posicionamiento float de la caja 1

Cuando se posiciona una caja de forma flotante: * La caja deja de pertenecer al flujo normal de la página, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante. * La caja flotante se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente.



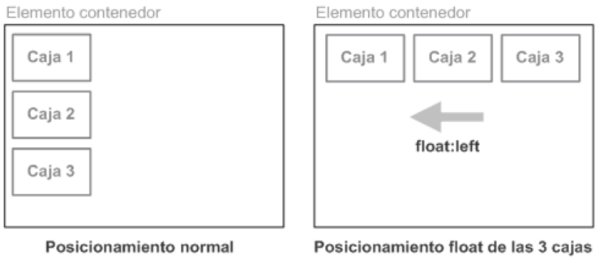
Posicionamiento normal
Posicionamiento float de la caja 1

Si en el anterior ejemplo la caja 1 se posiciona de forma flotante hacia la izquierda, el resultado es el que muestra la imagen a la izquierda.

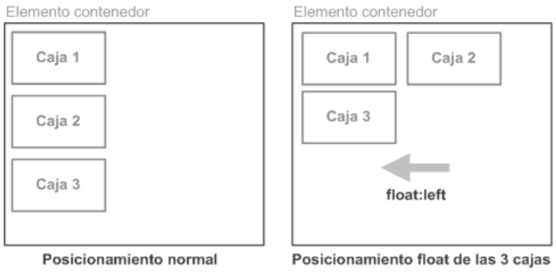
La caja 1 es de tipo flotante, por lo que desaparece del flujo normal de la página y el resto de cajas ocupan su lugar. El resultado es que la caja 2 ahora se muestra donde estaba la caja 1 y la caja 3 se muestra donde estaba la caja 2.

Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos.

Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible. En la imagen a la derecha se posicionan de forma flotante hacia la izquierda las tres cajas.



Posicionamiento normal
Posicionamiento float de las 3 cajas



Posicionamiento normal
Posicionamiento float de las 3 cajas

En el ejemplo anterior, las cajas no se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Como la caja 1 ya estaba posicionada lo más a la izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

Si no existe sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea, como aparece en la imagen superior.

Las cajas flotantes influyen en la disposición de todas las demás cajas. Los elementos en línea hacen sitio a las cajas flotantes adaptando su anchura al espacio libre dejado por la caja desplazada. Los elementos de bloque no les hacen sitio, pero sí que adaptan sus contenidos para que no se solapen con las cajas flotantes.

La propiedad CSS que permite posicionar de forma flotante una caja se denomina float:

Propiedad	**float**
Valores	left | right | none | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el tipo de posicionamiento flotante del elemento

Si se indica un valor left, la caja se desplaza hasta el punto más a la izquierda posible en esa misma línea (si no existe sitio en esa línea, la caja baja una línea y se muestra lo más a la izquierda posible en esa nueva línea). El resto de elementos adyacentes se adaptan y fluyen alrededor de la caja flotante.

El valor right tiene un funcionamiento idéntico, salvo que en este caso, la caja se desplaza hacia la derecha. El valor none permite anular el posicionamiento flotante de forma que el elemento se muestre en su posición original.

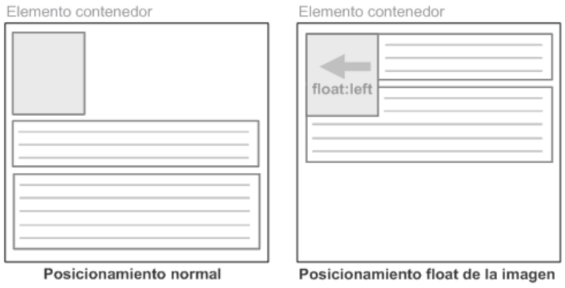
Ejercicio6

A partir del código HTML proporcionado **`Ejercicio6.html`**:

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejercicio posicionamiento float</title>
  <style type="text/css">

  </style>
</head>
<body>
    <section>
      <div>
         &laquo; Anterior   Siguiente &raquo;
      </div>
    </section>
</body>
</html>
```

« Anterior Siguiente »

Determinar las reglas CSS necesarias para que el resultado sea similar al mostrado en la siguiente imagen:



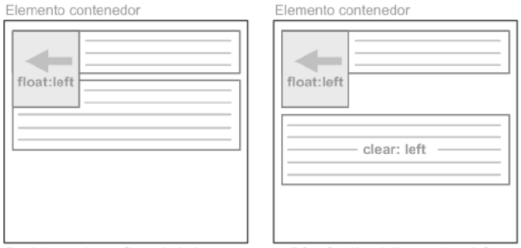
Los elementos que se encuentran alrededor de una caja flotante adaptan sus contenidos para que fluyan alrededor del elemento posicionado:

La regla CSS que se aplica en la imagen del ejemplo anterior es:

```
img {
  float: left;
}
```

Uno de los principales motivos para la creación del posicionamiento float fue precisamente la posibilidad de colocar imágenes alrededor de las cuales fluye el texto.

CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante float. De hecho, en muchas ocasiones es admisible que algunos contenidos fluyan alrededor de una imagen, pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen (como por ejemplo este gráfico a la izquierda).



Posicionamiento float de la imagen Párrafo "limpio" con clear:left

La propiedad clear permite modificar el comportamiento por defecto del posicionamiento flotante para forzar a un elemento a mostrarse debajo de cualquier caja flotante. La regla CSS que se aplica al segundo párrafo del ejemplo anterior es la siguiente:

```
<p style="clear: left;">...</p>
```

La definición formal de la propiedad clear se muestra a continuación:

Propiedad	**clear**
Valores	none | left | right | both | inherit
Se aplica a	Todos los elementos de bloque
Valor inicial	none
Descripción	Indica el lado del elemento que no debe ser adyacente a ninguna caja flotante

La propiedad clear indica el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante. Si se indica el valor left, el elemento se desplaza de forma descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante en el lado izquierdo.

La especificación oficial de CSS explica este comportamiento como "un desplazamiento descendente hasta que el borde superior del elemento esté por debajo del borde inferior de cualquier elemento flotante hacia la izquierda".

Si se indica el valor right, el comportamiento es análogo, salvo que en este caso se tienen en cuenta los elementos desplazados hacia la derecha. El valor both despeja los lados izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante hacia la izquierda o hacia la derecha.

Como se verá más adelante, la propiedad clear es imprescindible cuando se crean las estructuras de las páginas web complejas. Ejemplo (en primer lugar incorrecto): **Ej05.07a-Flotante.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejercicio posicionamiento float</title>
<style type="text/css">
      #paginacion {  border: 1px solid #CCC;
                     background-color: #E0E0E0;     padding: .5em;        }
       .derecha   { float: right; }
      .izquierda { float: left;  }
</style>
</head>
<body>
   <section>
      <div id="paginacion">
        <span class="izquierda">&laquo; Anterior</span>
        <span class="derecha">Siguiente &raquo;</span>
      </div>
</section>
</body> </html>
```

« Anterior Siguiente »

Los elementos Anterior y Siguiente se salen de su elemento contenedor y el resultado es visualmente incorrecto. El motivo de este comportamiento es que un elemento posicionado de forma flotante ya no pertenece al flujo normal de la página HTML. Por tanto, el elemento en realidad no encierra ningún contenido y por eso se visualiza incorrectamente.

La solución consiste en utilizar la propiedad overflow (que se explica más adelante) sobre el elemento contenedor: **Ej05.07b-FlotanteCorrecto.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Ejercicio posicionamiento float</title>

<style type="text/css">
#paginacion {
  border: 1px solid #CCC;
  background-color: #E0E0E0;
  padding: .5em;
  overflow: hidden;
}

.derecha   { float: right; }
.izquierda { float: left;  }

</style>
</head>

<body>
   <section>
      <div id="paginacion">
        <span class="izquierda">&laquo; Anterior</span>
        <span class="derecha">Siguiente &raquo;</span>
      </div>
</section>
</body> </html>
```

El resultado en el navegador sería el siguiente:

« Anterior Siguiente »

5.8 Visualización

Además de las propiedades que controlan el posicionamiento de los elementos, CSS define otras cuatro propiedades para controlar su visualización: display, visibility, overflow y z-index.

Utilizando algunas de estas propiedades es posible ocultar y/o hacer invisibles las cajas de los elementos, por lo que son imprescindibles para realizar efectos avanzados y animaciones.

5.8.1 Propiedades display y visibility

Las propiedades display y visibility controlan la visualización de los elementos. Las dos propiedades permiten ocultar cualquier elemento de la página. Habitualmente se utilizan junto con JavaScript para crear efectos dinámicos como mostrar y ocultar determinados textos o imágenes cuando el usuario pincha sobre ellos.

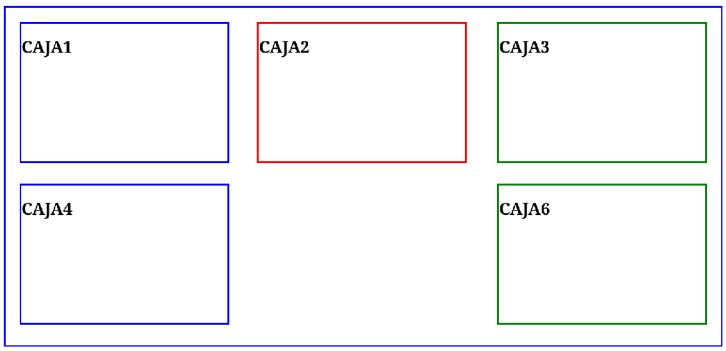
La propiedad display permite ocultar completamente un elemento haciendo que desaparezca de la página. Como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.

Por otra parte, la propiedad visibility permite hacer invisible un elemento, lo que significa que el navegador crea la caja del elemento pero no la muestra. En este caso, el resto de elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

Veamos un ejemplo de visibility: hidden, con 6 cajas uniformemente repartidas:

Ej05.08a-VisibilityHidden.html

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Fondos</title>
  <style type="text/css">
   #contenedor { width: 1000px;      height: 450px;
                 padding: 20px;      border: 3px solid blue;
   }

   #caja1 {      width: 30%;         height: 200px;
                 border: 3px solid blue;    float: left;
   }

   #caja2 {      width: 30%;         height: 200px;
                 float: left;        margin-left: 4%;
                 border: 3px solid red;
   }

   #caja3 {      width: 30%;         height: 200px;
                 border: 3px solid green;   float: right;
   }

   #caja4 {      width: 30%;         height: 200px;
                 float: left;        margin: 3.5% 4% 0 0;
                 border: 3px solid blue;
    }

   #caja5 {      width: 30%;         height: 200px;
                 float: left;        margin: 3.5% 4% 0 0;
                 border: 3px solid red;
                 visibility: hidden; /* Si comentamos esta línea aparece la caja*/
   }

   #caja6 {      width: 30%;         height: 200px;
                 margin-top: 3.5%;   float: left;
                 border: 3px solid green;
   }
  </style>
</head>
<body>
    <section>
      <div id="contenedor">
      <div id="caja1">
         <h1>CAJA1</h1></div>
      <div id="caja2">
         <h1>CAJA2</h1></div>
      <div id="caja3">
         <h1>CAJA3</h1></div>
      <div id="caja4">
         <h1>CAJA4</h1></div>
      <div id="caja5">
         <h1>CAJA5</h1></div>
      <div id="caja6">
         <h1>CAJA6</h1></div>
      </div>
    </section>
</body>
</html>
```



En general, cuando se oculta un elemento no es deseable que siga ocupando sitio en la página, por lo que la propiedad display se utiliza mucho más que la propiedad visibility.

A continuación se muestra la definición completa de la propiedad display:

Propiedad	display
Valores	inline | block | none | list-item | run-in | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | inherit
Se aplica a	Todos los elementos
Valor inicial	inline
Descripción	Permite controlar la forma de visualizar un elemento e incluso ocultarlo

Las posibilidades de la propiedad display son mucho más avanzadas que simplemente ocultar elementos. En realidad, la propiedad display modifica la forma en la que se visualiza un elemento.

Los valores más utilizados son inline, block y none. El valor block muestra un elemento como si fuera un elemento de bloque, independientemente del tipo de elemento que se trate. El valor inline visualiza un elemento en forma de elemento en línea, independientemente del tipo de elemento que se trate.

El valor none oculta un elemento y hace que desaparezca de la página. El resto de elementos de la página se visualizan como si no existiera el elemento oculto, es decir, pueden ocupar el espacio en el que se debería visualizar el elemento.

El siguiente ejemplo muestra el uso de la propiedad display para mostrar un elemento de bloque como si fuera un elemento en línea y para mostrar un elemento en línea como si fuera un elemento de bloque: **Ej05.08b-Display.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>InlineBlock</title>
  <style type="text/css">
      div {border: 1px solid blue;}
      #inline {   display:inline; }
      a {   border: 1px solid red;}
      #block { display:block; }
  </style>
</head>
<body>
    <section>
      <div>DIV normal</div> <br />
      <div id="inline">DIV con display:inline</div> <br /><br />
      <a href="#">Enlace normal</a><br /><br />
      <a id="block" href="#">Enlace con display:block</a>
    </section>
</body>
</html>
```

Como se verá más adelante, la propiedad display: inline se puede utilizar en las listas (<ul>, <ol>) que se quieren mostrar horizontalmente y la propiedad display: block se emplea frecuentemente para los enlaces que forman el menú de navegación.

La propiedad display es una de las propiedades CSS más infrautilizadas. Aunque todos los diseñadores conocen esta propiedad y utilizan sus valores inline, block y none, las posibilidades de display son mucho más avanzadas.

De hecho, la propiedad display es una de las más complejas de CSS 2.1, ya que establece el tipo de la caja que genera cada elemento. La propiedad display es tan compleja que casi ningún navegador es capaz de mostrar correctamente todos sus valores.

Uno de los valores más curiosos de display es inline-block, que crea cajas que son de bloque y en línea de forma simultánea. Una caja de tipo inline-block se comporta como si fuera de bloque, pero respecto a los elementos que la rodean es una caja en línea.
El enlace del siguiente ejemplo es de tipo inline-block, lo que permite por ejemplo establecerle un tamaño mediante la propiedad width **Ej05.08c-InlineBlock.html:**

```
<!DOCTYPE html>
<html lang="es">
<head>      <!-- Ej05.08c-InlineBlock.html -->
   <meta charset="UTF-8" />
   <title>InlineBlock</title>
   <style>
      p {border: 2px solid red;
         padding: 3px;}
      a {         display: inline-block;
               /*display: inline;     PROBAR!*/
               /*display: block;      PROBAR!*/
               width: 150px;          padding: 3px;
               margin: 5px;           border: 2px solid blue;
               text-align: center;    }
   </style>
</head>
<body>
      <section>
         <p> El mercado de la mensajería no puede estar más fragmentado ahora mismo.
            Tenemos un líder, <a href="http://www.whatsapp.com">WhatsApp</a>;
            pero también tenemos LINE, Facebook Poke, Viber, Skype, GroupMe, iMessage… la
            única compañía que se ha quedado atrás en esta revolución de la mensajería
            instantánea es Google. Pero no por mucho tiempo, porque parece que la compañía
            trabaja para replantear todos sus servicios de chat hacia un nuevo servicio
            único. </p>
      </section>
</body>
</html>
```

Otro de los valores definidos por la propiedad display es list-item, que hace que cualquier elemento de cualquier tipo se muestre como si fuera un elemento de una lista (elemento <li>). El siguiente ejemplo muestra tres párrafos que utilizan la propiedad display: list-item para simular que son una lista de elementos de tipo <ul>:**Ej05.08d-ListItem.html:**

```
<!DOCTYPE html>
<html lang="es">
<head>      <!-- Ej05.08d-ListItem.html -->
   <meta charset="UTF-8" />
   <title>List Item</title>
   <style>
      p {   display: list-item;
            margin: 2em;
            /*list-style-type: hiragana;*/}
   </style>
</head>
<body>
      <section>
         <p> Si hace algunos meses se sumergió en atolones para permitirnos
         descubrir la belleza de explorar el mar como un buceador,
         ahora Google nos pone en la piel de un caminante en un escenario de ensueño. </p>
         <p> Y lo hace gracias a un artilugio en forma de mochila que tiene 15 cámaras
         colocadas en la parte de arriba que permite obtener imágenes en 360 grados. </p>
      </section>
</body>
</html>
```

Los elementos con la propiedad display: list-item son exactamente iguales que los elementos <li> a efectos de su visualización, por lo que se pueden utilizar las propiedades de listas como list-style-type, list-style-image, list-style-position y list-style.
Uno de los valores más curiosos de la propiedad display es run-in, que genera una caja de bloque o una caja en línea dependiendo del contexto, es decir, dependiendo de sus elementos adyacentes.

El comportamiento de las cajas de tipo run-in se rige por las siguientes reglas:

- Si la caja run-in contiene una caja de bloque, se convierte en una caja de bloque.
- Si después de la caja run-in se encuentra una caja de bloque (que no esté posicionada de forma absoluta y tampoco esté posicionada de forma flotante), la caja run-in se convierte en una caja en línea en el interior de la caja de bloque. *
- En cualquier otro caso, la caja run-in se convierte en una caja de bloque.

El siguiente ejemplo muestra una misma caja de tipo run-in que se visualiza de forma muy diferente en función del tipo de caja que existe a continuación **Ej05.08e-RunIn.html:**

NOTA IMPORTANTE: No funciona en Firefox (y si en Chromium)

```
<!DOCTYPE html> <html lang="es">
<head>      <!-- Ej05.08e-RunIn.htmll -->
   <meta charset="UTF-8" />
   <title>List Item</title>
   <style>
      .run-in {   display: run-in;   border: 2px solid blue;       }
      #p1   {     display: block;    border: 2px solid red;                }
      #p2   {     display: inline;   border: 2px solid green;      }
      h3    {     display: run-in;    box-shadow: 5px 5px 5px blue;       }
   </style>
</head>
<body>
         <p class="run-in"> <strong>[Display: run-in]</strong> Párrafo Inicial </p>
         <p id="p1">    <strong>[Display: block]</strong> Esto será un párrafo Bloque </p>
         <p class="run-in"> <strong>[Display: run-in]</strong> Otro Párrafo</p>
         <p id="p2">    <strong>[Display: inline]</strong> Esto es un párrafo en Línea</p>
         <h3> Titulo Secundario</h3>
         <p> Un potente terremoto de ocho grados de magnitud en la escala de Richter
            ha sacudido el este de las Islas Salomón en el Pacífico, activando
            la alerta de tsunami en el Centro de Alerta en Hawai. </p>
</body></html>
```

El estándar CSS 2.1 incluye un ejemplo del posible uso del valor run-in. En este ejemplo, un título de sección <h3> crea una caja run-in, de forma que cuando va seguido de un párrafo, el titular se mete dentro del párrafo (como se indica en el ejemplo anterior en negrita).

El resto de valores de la propiedad display están relacionados con las tablas y hacen que un elemento se muestre como si fuera una parte de una tabla: fila, columna, celda o grupos de filas/columnas. Los valores definidos por la propiedad display son inline-table, table-row-group, table-header-group, table-footer-group, table-row, table-column-group, table-column, table-cell, table-caption.

Aunque los valores relacionados con las tablas son los más avanzados, también son los que peor soportan los navegadores. A continuación se muestra un ejemplo con tres párrafos de texto que establecen la propiedad display: table-cell. Ejemplo **Ej05.08f-TableCell.html:**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Fondos</title>
  <style type="text/css">
      p     {  display: table-cell; padding: 10px;  }
      span  {  font-weight: bold;    }
  </style> </head>
<body> <section>
<p><span>[display: table-cell]</span> Lorem ipsum dolor sit amet, consectetuer adipiscing
elit.Maecenas non tortor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices
posuere cubilia Curae;Sed fermentum lorem a velit.</p>
<p><span>[display: table-cell]</span> In molestie suscipit libero. Cras sem. Nunc non
tellus et urna mattis tempor. Nulla nec tellus a quam hendrerit venenatis. Suspendisse
pellentesque odio et est. Morbi sed nisl sed dui consequat sodales.</p>
<p><span>[display: table-cell]</span>Morbi sed nisl sed dui consequat sodales. Vivamus
ornare felis nec est. Phasellus massa justo, ornare sed, malesuada a, dignissim a, nibh.
Vestibulum vitae nunc at lectus euismod feugiat.Nullam eleifend. </p>
    </section>
</body></html>
```

La propiedad display: table-cell hace que cualquier elemento se muestre como si fuera una celda de una tabla. Como en el ejemplo anterior los tres elementos <p> utilizan la propiedad display: table-cell, el resultado es visualmente idéntico a utilizar una tabla y tres elementos <td>.

Como los valores relacionados con las tablas hacen que cualquier elemento que no sea una tabla se muestre y comporte como si lo fuera, se pueden utilizar para crear los layouts de las páginas. Hace años, la estructura de las páginas se definía mediante tablas, filas y columnas. Esta solución tiene innumerables desventajas y por eso todos los diseñadores web profesionales crean la estructura de sus páginas mediante CSS y elementos .

No obstante, las tablas tienen algunas ventajas en su comportamiento respecto a los elementos posicionados de forma absoluta o flotante. La principal ventaja es que todas las celdas de una fila siempre tienen la misma altura, por lo que si se utilizan tablas no se sufre el problema de las columnas de página con diferente altura.

Además, la estructura creada con una tabla nunca se rompe, ya que las celdas de datos nunca se visualizan una debajo de otra cuando la ventana del navegador se hace muy pequeña. Sin embargo, cuando se define la estructura mediante elementos posicionados es posible que la página se rompa y alguna columna se muestre debajo de otros contenidos.

Utilizando la propiedad display de forma avanzada es posible crear una estructura de página que sea semánticamente correcta, esté diseñada exclusivamente con CSS y que se comporte exactamente igual que como lo hacen las tablas. El siguiente código HTML corresponde a la estructura de una página con tres columnas: Ejemplo **Ej05.08g-3Columnas.html:**

```
<!DOCTYPE html> <html lang="es">            <!-- Ej05.08g-3Columnas.html -->
  <head> <meta charset="UTF-8" /> <title>3 Columnas</title>
  <style type="text/css">
   #contenedor {     display: table;
                     border-spacing: 10px;   }
   #contenidos {     display: table-row;     }

   #izquierda, #principal, #derecha {
                     display: table-cell;
                     border: 2px solid blue;
                     padding: 15px;          }
   #principal {               width: 60%; }
   #izquierda, #derecha {     width: 20%;
                              border: 5px solid blue; }
  </style>
</head> <body>
      <div id="contenedor">
         <div id="contenidos">
            <div id="izquierda">
               <h3>El año de Linux en los bolsillos </h3>
            <p> El primer mes de 2013 aportó varias novedades interesantes que abren nuevas
            oportunidades y opciones a usuarios, desarrolladores y sistemas móviles.</p>
            </div>
            <div id="principal">
               <h3>Wine para Android permitira ejecutar aplicaciones de Windows</h3>
               <p>Después de tratar con algunos problemas iniciales en su demo, Julliard
               tranquilamente mostró Wine corriendo en Android. Sin embargo, el rendimiento
               fue terriblemente lento. Los problemas de rendimiento, aunque se atribuyeron
               a ejecutar el entorno emulado Android en lugar de mostrar la aplicación Wine
               desde un dispositivo nativo. </p>
            </div>
            <div id="derecha">
               <h3>Ubuntu ya representa el 1,12% del uso de Steam </h3>
               <p>Valve ha publicado el informe mensual de Steam en el que se repasan,
               entre otras, las estadísticas de cuota según sistema operativo</p>
            </div>
         </div>
      </div>
</body> </html>
```

El elemento #contenedor se visualiza como una tabla porque se le aplica la propiedad display: table. De esta forma, se pueden aplicar al elemento #contenedor propiedades exclusivas de las tablas como border-spacing. El elemento #contenidos se visualiza como si fuese una fila de tabla (etiqueta <tr>). En su interior se encuentran las tres columnas de la página que se visualizan como si fueran tres elementos <td> gracias a la propiedad display: table-cell.

La estructura de la página del ejemplo anterior está diseñada exclusivamente con CSS pero se comporta como si fuera una tabla. Todas las columnas de la página tienen la misma altura sin necesidad de recurrir a ningún truco y la página nunca se rompe por muy pequeña que se haga la ventana del navegador.

Por último, aunque el estándar CSS 2.1 establece que el valor por defecto de la propiedad display es inline, todos los navegadores obvian esta recomendación y asignan por defecto el valor block a los elementos de bloque y el valor inline a los elementos en línea.

Por su parte, la definición completa de la propiedad visibility es mucho más sencilla:

Propiedad	**visibility**
Valores	visible | hidden | collapse | inherit
Se aplica a	Todos los elementos
Valor inicial	visible
Descripción	Permite hacer visibles e invisibles a los elementos

Las posibilidades de la propiedad visibility son mucho más limitadas que las de la propiedad display, ya que sólo permite hacer visibles o invisibles a los elementos de la página.

Inicialmente todas las cajas que componen la página son visibles. Empleando el valor hidden es posible convertir una caja en invisible para que no muestre sus contenidos. El resto de elementos de la página se muestran como si la caja todavía fuera visible, por lo que en el lugar donde originalmente se mostraba la caja invisible, ahora se muestra un hueco vacío.

or último, el valor collapse de la propiedad visibility sólo se puede utilizar en las filas, grupos de filas, columnas y grupos de columnas de una tabla. Su efecto es similar al de la propiedad display, ya que oculta completamente la fila y/o columna y se pueden mostrar otros contenidos en ese lugar. Si se utiliza el valor collapse sobre cualquier otro tipo de elemento, su efecto es idéntico al valor hidden.

5.8.2 Relación entre display, float y position

Cuando se establecen las propiedades display, float y position sobre una misma caja, su interpretación es la siguiente:

- Si display vale none, se ignoran las propiedades float y position y la caja no se muestra en la página.
- Si position vale absolute o fixed, la caja se posiciona de forma absoluta, se considera que float vale none y la propiedad display vale block tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades top, right, bottom y left.
- En cualquier otro caso, si float tiene un valor distinto de none, la caja se posiciona de forma flotante y la propiedad display vale block tanto para los elementos en línea como para los elementos de bloque.

5.8.3 La propiedad Overflow

Normalmente, los contenidos de un elemento se pueden mostrar en el espacio reservado para ese elemento. Sin embargo, en algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda.

La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad width y/o height. Otra situación habitual es la de las líneas muy largas contenidas dentro de un elemento <pre>, que hacen que la página entera sea demasiado ancha.

CSS define la propiedad overflow para controlar la forma en la que se visualizan los contenidos que sobresalen de sus elementos.

Propiedad	**overflow**
Valores	visible | hidden | scroll | auto | inherit
Se aplica a	Elementos de bloque y celdas de tablas
Valor inicial	visible
Descripción	Permite controlar los contenidos sobrantes de un elemento

Los valores de la propiedad overflow tienen el siguiente significado:

- visible: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.
- hidden: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.
- scroll: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de scroll que permiten visualizar el resto del contenido.
- auto: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad scroll.

Ejemplo **Ej05.08h-Overflow.html:**

```
<!DOCTYPE html><html lang="es">      <!-- Ej05.08h-Overflow.html -->
  <head> <meta charset="UTF-8" />    <title>Overflow</title>
  <style type="text/css">
   div {       display: inline;              float: left;
            width: 20%;                      padding: 20px;
            margin: 0 10px;          height: 100px;
            border: 2px solid blue;  }
   #principal  {  overflow: hidden;  }
   #derecha    {  overflow: scroll;  overflow-x: hidden;   }
  </style> </head>
<body>
      <div id="izquierda">
         <h3>El año de Linux en los bolsillos </h3>
         <p> El primer mes de 2013 aportó varias novedades interesantes que abren
         nuevas oportunidades y opciones a usuarios, desarrolladores y sistemas
         móviles basados en Linux.</p>
      </div>
      <div id="principal">
         <h3>Wine para Android permitira ejecutar aplicaciones de Windows</h3>
         <p>Después de tratar con algunos problemas iniciales en su demo, Julliard
         tranquilamente mostró Wine corriendo en Android. Sin embargo, el rendimiento
         fue terriblemente lento. </p>
      </div>
      <div id="derecha">
         <h3>Ubuntu ya representa el 1,12% del uso de Steam </h3>
         <p>Valve ha publicado el informe mensual de Steam en el que se repasan,
         entre otras, las estadísticas de cuota según sistema operativo</p>
      </div>
</body> </html>
```

5.8.4 Propiedad z-index

Además de posicionar una caja de forma horizontal y vertical, CSS permite controlar la posición tridimensional de las cajas posicionadas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional de un elemento se establece sobre un tercer eje llamado Z y se controla mediante la propiedad z-index. Utilizando esta propiedad es posible crear páginas complejas con varios niveles o capas.

A continuación se muestra la definición formal de la propiedad z-index:

Propiedad	**z-index**
Valores	auto | <numero> | inherit
Se aplica a	Elementos que han sido posicionados explícitamente
Valor inicial	auto
Descripción	Establece el nivel tridimensional en el que se muestra el elemento

El valor más común de la propiedad z-index es un número entero. Aunque la especificación oficial permite los números negativos, en general se considera el número 0 como el nivel más bajo.

Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja. Un elemento con z-index: 10 se muestra por encima de los elementos con z-index: 8 o z-index: 9, pero por debajo de elementos con z-index: 20 o z-index: 50. Ejemplo **Ej05.08i-Capas.html:**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Fondos</title>
  <style type="text/css">
   div { border: 2px solid blue;
         padding: 15px;              width: 30%;}

   #caja1 { position: absolute;
            top: 100px; left: 100px;
            background-color: aquamarine;
            z-index: 2;}
   #caja2 { position: absolute;
            top: 130px; left: 130px;
            background-color: khaki;
            z-index: 3;}
   #caja3 { position: absolute;
            top: 160px; left: 160px;
            background-color: pink;
            z-index: 1;}
  </style>
</head>
<body>
    <section>
<div id="caja1">Caja 1 - Caja 1 - Caja 1 - Caja 1 </div>
<div id="caja2">Caja 2 - Caja 2 - Caja 2 - Caja 2 </div>
<div id="caja3">Caja 3 - Caja 3 - Caja 3 - Caja 3 </div>
    </section>
</body>
</html>
```

La propiedad z-index sólo tiene efecto en los elementos posicionados, por lo que es obligatorio que la propiedad z-index vaya acompañada de la propiedad position. Si debes posicionar un elemento pero no quieres moverlo de su posición original ni afectar al resto de elementos de la página, puedes utilizar el posicionamiento relativo (position: relative).

6 TEXTO

6.1 Tipografía

CSS define numerosas propiedades para modificar la apariencia del texto. A pesar de que no dispone de tantas posibilidades como los lenguajes y programas específicos para crear documentos impresos, CSS permite aplicar estilos complejos y muy variados al texto de las páginas web.

La propiedad básica que define CSS relacionada con la tipografía se denomina color y se utiliza para establecer el color de la letra.

Propiedad	**color**
Valores	<color> | inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el color de letra utilizado para el texto

Aunque el color por defecto del texto depende del navegador, todos los navegadores principales utilizan el color negro. Para establecer el color de letra de un texto, se puede utilizar cualquiera de las cinco formas que incluye CSS para definir un color.

A continuación se muestran varias reglas CSS que establecen el color del texto de diferentes formas:

```
h1 { color: #369; }
p  { color: black; }
a, span { color: #B1251E; }
div { color: rgb(71, 98, 176); }
```

Como el valor de la propiedad color se hereda, normalmente se establece la propiedad color en el elemento body para establecer el color de letra de todos los elementos de la página:

```
body { color: #777; }
```

En el ejemplo anterior, todos los elementos de la página muestran el mismo color de letra salvo que establezcan de forma explícita otro color. La única excepción de este comportamiento son los enlaces que se crean con la etiqueta <a>. Aunque el color de la letra se hereda de los elementos padre a los elementos hijo, con los enlaces no sucede lo mismo, por lo que es necesario indicar su color de forma explícita:

```
/* Establece el mismo color a todos los elementos de
   la página salvo los enlaces */
body { color: #777; }

/* Establece el mismo color a todos los elementos de
   la página, incluyendo los enlaces */
body, a { color: #777; }
```

La otra propiedad básica que define CSS relacionada con la tipografía se denomina font-family y se utiliza para indicar el tipo de letra con el que se muestra el texto.

Propiedad	**font-family**
Valores	((<nombre_familia> | <familia_generica>) (,<nombre_familia> | <familia_generica>)*) | inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el tipo de letra utilizado para el texto

El tipo de letra del texto se puede indicar de dos formas diferentes:

- Mediante el nombre de una familia tipográfica: en otras palabras, mediante el nombre del tipo de letra, como por ejemplo "Arial", "Verdana", "Garamond", etc.
- Mediante el nombre genérico de una familia tipográfica: los nombres genéricos no se refieren a ninguna fuente en concreto, sino que hacen referencia al estilo del tipo de letra. Las familias genéricas definidas son serif (tipo de letra similar a Times New Roman), sans-serif (tipo Arial), cursive (tipo Comic Sans), fantasy (tipo Impact) y monospace (tipo Courier New).

Los navegadores muestran el texto de las páginas web utilizando los tipos de letra instalados en el ordenador o dispositivo del propio usuario. De esta forma, si el diseñador indica en la propiedad font-family que el texto debe mostrarse con un tipo de letra especialmente raro o rebuscado, casi ningún usuario dispondrá de ese tipo de letra.

Para evitar el problema común de que el usuario no tenga instalada la fuente que quiere utilizar el diseñador, CSS permite indicar en la propiedad font-family más de un tipo de letra. El navegador probará en primer lugar con el primer tipo de letra indicado. Si el usuario la tiene instalada, el texto se muestra con ese tipo de letra.

Si el usuario no dispone del primer tipo de letra indicado, el navegador irá probando con el resto de tipos de letra hasta que encuentre alguna fuente que esté instalada en el ordenador del usuario. Evidentemente, el diseñador no puede indicar para cada propiedad font-family tantos tipos de letra como posibles fuentes parecidas existan.

Para solucionar este problema se utilizan las familias tipográficas genéricas. Cuando la propiedad font-family toma un valor igual a sans-serif, el diseñador no indica al navegador que debe utilizar la fuente Arial, sino que debe utilizar "la fuente que más se parezca a Arial de todas las que tiene instaladas el usuario".

Por todo ello, el valor de font-family suele definirse como una lista de tipos de letra alternativos separados por comas. El último valor de la lista es el nombre de la familia tipográfica genérica que más se parece al tipo de letra que se quiere utilizar.

Un inciso: vamos a realizar un menú con un elemento nuevo, *a:hover*, **Ej06.01a-Menu.html:**

```
<!DOCTYPE html><html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Fondos</title>
  <style type="text/css">
      #menu    {  padding: 0;         }
      #menu li {  display: inline;    }

      #menu li a {   font-family: Arial;            font-size:11px;
                     color: #fff;                   text-decoration: none;
                     padding: 10px;                 float:left;
                     background-color: #2175bc;     }

      #menu li a:hover {      margin-top:-2px;      background-color: #2586d7;
                              padding-bottom:12px;  }
  </style>
</head>
<body>
    <section>
<ul id="menu">
      <li><a href="#">Portada</a></li>      <li><a href="#">Sobre mi</a></li>
      <li><a href="#">Servicios</a></li>    <li><a href="#">Clientes</a></li>
      <li><a href="#">Productos</a></li>    <li><a href="#">Preguntas</a></li>
      <li><a href="#">Ayuda</a></li>        <li><a href="#">Contacto</a></li>
</ul>
    </section>
</body> </html>
```

Las listas de tipos de letra más utilizadas son las siguientes:

```
font-family: Arial, Helvetica, sans-serif;
font-family: "Times New Roman", Times, serif;
font-family: "Courier New", Courier, monospace;
font-family: Georgia, "Times New Roman", Times, serif;
font-family: Verdana, Arial, Helvetica, sans-serif;
```

Ya que las fuentes que se utilizan en la página deben estar instaladas en el ordenador del usuario, cuando se quiere disponer de un diseño complejo con fuentes muy especiales, se debe recurrir a soluciones alternativas.

La solución más sencilla consiste en crear imágenes en las que se muestra el texto con la fuente deseada. Esta técnica solamente es viable para textos cortos (por ejemplo los titulares de una página) y puede ser manual (creando las imágenes una por una) o automática, utilizando JavaScript, PHP y/o CSS.

Otra alternativa es la de la sustitución automática de texto basada en Flash. La técnica más conocida es la de sIFR, de la que se puede encontrar más información en http://wiki.novemberborn.net/sifr

Una vez seleccionado el tipo de letra, se puede modificar su tamaño con la propiedad font-size.

propiedad	**font-size**
Valores	<tamaño_absoluto> | <tamaño_relativo> | <medida> | <porcentaje> | inherit
Se aplica a	Todos los elementos
Valor inicial	medium
Descripción	Establece el tamaño de letra utilizado para el texto

Además de todas las unidades de medida relativas y absolutas y el uso de porcentajes, CSS permite utilizar una serie de palabras clave para indicar el tamaño de letra del texto:

- tamaño_absoluto: indica el tamaño de letra de forma absoluta mediante alguna de las siguientes palabras clave: xx-small, x-small, small, medium, large, x-large, xx-large.
- tamaño_relativo: indica de forma relativa el tamaño de letra del texto mediante dos palabras clave (larger, smaller) que toman como referencia el tamaño de letra del elemento padre.

Veamos un ejemplo donde ver los distintos tamaños: **Ej06.01b-Fuentes.html:**

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Fondo Position</title>
   <style>
      #contenedor { width: 1250px; }

      #caja1, #caja2, #caja3, #caja4, #caja5 {
            padding: 10px;
            width: 200px;
            height: 500px;
            border: 5px solid blue;
            float: left;      }

      #caja2, #caja4 {  margin: 0px 20px;    }
   </style>
</head>
```

```
<body>
      <section>
         <div id="contenedor">
            <div id="caja1">  <h1>Absoluto</h1>
<p style="font-size: x-small">x-small</p>   <p style="font-size: small">small</p>
<p style="font-size: medium">medium</p>     <p style="font-size: large">large</p>
<p style="font-size: x-large">x-large</p>   <p style="font-size: xx-large">xx-large</p>
            </div>

            <div id="caja2"><h1>Em</h1>
<p style="font-size: 0.7em">0.7em</p>       <p style="font-size: 0.9em">0.9em</p>
<p style="font-size: 1em">1em</p>           <p style="font-size: 1.5em">1.5em</p>
<p style="font-size: 2em">2em</p>           <p style="font-size: 3em">3em</p>
            </div>
            <div id="caja3"><h1>Porcentaje</h1>
<p style="font-size: 50%">50%</p>           <p style="font-size: 75%">75%</p>
<p style="font-size: 100%">100%</p>         <p style="font-size: 150%">150%</p>
<p style="font-size: 200%">50%</p>          <p style="font-size: 300%">300%</p>
         </div>

            <div id="caja4"><h1>Pixeles</h1>
<p style="font-size: 8px">8px</p>           <p style="font-size: 12px">12px</p>
<p style="font-size: 18px">18px</p>         <p style="font-size: 24px">24px</p>
<p style="font-size: 36px">36px</p>         <p style="font-size: 48px">48px</p>
         </div>

            <div id="caja5"><h1>Puntos</h1>
<p style="font-size: 8pt">8pt</p>           <p style="font-size: 12pt">12pt</p>
<p style="font-size: 18pt">18pt</p>         <p style="font-size: 24pt">24pt</p>
<p style="font-size: 36pt">36pt</p>         <p style="font-size: 48pt">48pt</p>
         </div>
      </div>
   </section>
</body>
</html>
```

CSS recomienda indicar el tamaño del texto en la unidad em o en porcentaje (%). Además, es habitual indicar el tamaño del texto en puntos (pt) cuando el documento está específicamente diseñado para imprimirlo.

Por defecto los navegadores asignan los siguientes tamaños a los títulos de sección: <h1> = xx-large, <h2> = x-large, <h3> = large, <h4> = medium, <h5> = small, <h6> = x-small.

Una vez indicado el tipo y el tamaño de letra, es habitual modificar otras características como su grosor (texto en negrita) y su estilo (texto en cursiva). La propiedad que controla la anchura de la letra es font-weight.

Propiedad	**font-weight**
Valores	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece la anchura de la letra utilizada para el texto

Los valores que normalmente se utilizan son normal (el valor por defecto) y bold para los textos en negrita. El valor normal equivale al valor numérico 400 y el valor bold al valor numérico 700.

Por defecto, los navegadores muestran el texto de los elementos en cursiva y el texto de los elementos en negrita. La propiedad font-weight permite alterar ese aspecto por defecto y mostrar por ejemplo los elementos como cursiva y negrita y los elementos destacados mediante un color de fondo y sin negrita.

Además de la anchura de la letra, CSS permite variar su estilo mediante la propiedad font-style.

Propiedad	**font-style**
Valores	normal | italic | oblique | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo de la letra utilizada para el texto

Normalmente la propiedad font-style se emplea para mostrar un texto en cursiva mediante el valor italic. El siguiente ejemplo muestra una aplicación práctica de las propiedades font-weight y font-style: **Ej06.01c-Resaltado.html:**

```
<!DOCTYPE html><html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Fondo Position</title>
   <style>
      #especial em {          font-weight: bold;      }
      #especial strong {      font-weight: normal;
                              background-color: #FFFF66;
                              padding: 2px;         }
   </style>
</head>
<body>
      <section>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut in
      purus ac <em>libero nonummy vestibulum</em>. Nullam molestie, nunc id
      nonummy laoreet, <strong>tortor diam mollis elit</strong>, quis hendrerit
      libero lorem vitae nunc.</p>

      <p id="especial">Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
      Ut in purus ac <em>libero nonummy vestibulum</em>. Nullam molestie, nunc id
      nonummy laoreet, <strong>tortor diam mollis elit</strong>, quis hendrerit
      libero lorem vitae nunc.</p>
      </section>
</body>
</html>
```

Por último, CSS permite otra variación en el estilo del tipo de letra, controlado mediante la propiedad font-variant.

Propiedad	**font-variant**
Valores	normal | small-caps | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo alternativo de la letra utilizada para el texto

La propiedad font-variant no se suele emplear habitualmente, ya que sólo permite mostrar el texto con letra versal (mayúsculas pequeñas).: **Ej06.01d-Versalitas.html:**

```
<!DOCTYPE html><html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Fondo Position</title>
   <style>
      p {   font-variant: small-caps;}
   </style>
</head>
<body>
      <section>
      <p>Ejemplo de Versalitas</p>
      </section>
</body></html>
```

Por otra parte, CSS proporciona una propiedad tipo *"shorthand"* denominada font y que permite indicar de forma directa algunas o todas las propiedades de la tipografía de un texto.

Propiedad	**font**
Valores	((<font-style> || <font-variant> || <font-weight>)? <font-size> (/ <line-height>)? <font-family>) | caption | icon | menu | message-box | small-caption | status-bar | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Permite indicar de forma directa todas las propiedades de la tipografía de un texto

El orden en el que se deben indicar las propiedades del texto es el siguiente:

- En primer lugar y de forma opcional se indican el font-style, font-variant y font-weight en cualquier orden.
- A continuación, se indica obligatoriamente el valor de font-size seguido opcionalmente por el valor de line-height.
- Por último, se indica obligatoriamente el tipo de letra a utilizar.

Ejemplos de uso de la propiedad font:

```
font: 76%/140% Verdana,Arial,Helvetica,sans-serif;
font: normal 24px/26px "Century Gothic","Trebuchet MS",Arial,Helvetica,sans-
serif;
font: normal .94em "Trebuchet MS",Arial,Helvetica,sans-serif;
font: bold 1em "Trebuchet MS",Arial,Sans-Serif;
font: normal 0.9em "Lucida Grande", Verdana, Arial, Helvetica, sans-serif;
font: normal 1.2em/1em helvetica, arial, sans-serif;
font: 11px verdana, sans-serif;
font: normal 1.4em/1.6em "helvetica", arial, sans-serif;
font: bold 14px georgia, times, serif;
```

Aunque su uso no es muy común, la propiedad font también permite indicar el tipo de letra a utilizar mediante una serie de palabras clave: caption, icon, menu, message-box, small-caption, status-bar.

Si por ejemplo se utiliza la palabra status-bar, el navegador muestra el texto con el mismo tipo de letra que la que utiliza el sistema operativo para mostrar los textos de la barra de estado de las ventanas. La palabra icon se puede utilizar para mostrar el texto con el mismo tipo de letra que utiliza el sistema operativo para mostrar el nombre de los iconos y así sucesivamente.

6.2 Texto

Además de las propiedades relativas a la tipografía del texto, CSS define numerosas propiedades que determinan la apariencia del texto en su conjunto. Estas propiedades adicionales permiten controlar al alineación del texto, el interlineado, la separación entre palabras, etc.

La propiedad que define la alineación del texto se denomina text-align.

Propiedad	**text-align**
Valores	left | right | center | justify | inherit
Se aplica a	Elementos de bloque y celdas de tabla
Valor inicial	left
Descripción	Establece la alineación del contenido del elemento

Los valores definidos por CSS permiten alinear el texto según los valores tradicionales: a la izquierda (left), a la derecha (right), centrado (center) y justificado (justify).

La propiedad text-align no sólo alinea el texto que contiene un elemento, sino que también alinea todos sus contenidos, como por ejemplo las imágenes.

El interlineado de un texto se controla mediante la propiedad line-height, que permite controlar la altura ocupada por cada línea de texto:

Propiedad	**line-height**
Valores	normal | <numero> | <medida> | <porcentaje> | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer la altura de línea de los elementos

Además de todas las unidades de medida y el uso de porcentajes, la propiedad line-height permite indicar un número sin unidades que se interpreta como el múltiplo del tamaño de letra del elemento. Por tanto, estas tres reglas CSS son equivalentes:

```
p { line-height: 1.2; font-size: 1em }
p { line-height: 1.2em; font-size: 1em }
p { line-height: 120%; font-size: 1em }
```

Siempre que se utilice de forma moderada, el interlineado mejora notablemente la legibilidad de un texto. Veamos un ejemplo con ambas propiedades, **Ej06.02a-AlineaInterlineado.html:**

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Fondo Position</title>
   <style>
      #p1 { text-align: center;      line-height: 1em;}
      #p2 { text-align: justify;     line-height: 2em;}
   </style>
</head>
<body>
      <section>
      <p id="p1">Los valores definidos por CSS permiten alinear el texto según los valores
      tradicionales: a la izquierda (left), a la derecha (right), centrado (center) y
      justificado (justify).</p>
      <p id="p2">La propiedad text-align no sólo alinea el texto que contiene un elemento,
      sino que también alinea todos sus contenidos, como por ejemplo las imágenes.</p>
      </section>
</body>
</html>
```

Además de la decoración que se puede aplicar a la tipografía que utilizan los textos, CSS define otros estilos y decoraciones para el texto en su conjunto. La propiedad que decora el texto se denomina text-decoration.

Propiedad	**text-decoration**
Valores	none | (underline || overline || line-through || blink) | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece la decoración del texto (subrayado, tachado, parpadeante, etc.)

Las distintas opciones son:

- El valor underline subraya el texto, por lo que puede confundir a los usuarios haciéndoles creer que se trata de un enlace.
- El valor overline añade una línea en la parte superior del texto, un aspecto que raramente es deseable.
- El valor line-through muestra el texto tachado con una línea continua, por lo que su uso tampoco es muy habitual.
- Por último, el valor blink muestra el texto parpadeante y se recomienda evitar su uso por las molestias que genera a la mayoría de usuarios.

Una de las propiedades de CSS más desconocidas y que puede ser de gran utilidad en algunas circunstancias es la propiedad text-transform, que puede variar de forma sustancial el aspecto del texto.

Propiedad	**text-transform**
Valores	capitalize | uppercase | lowercase | none | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Transforma el texto original (lo transforma a mayúsculas, a minúsculas, etc.)

La propiedad text-transform permite mostrar el texto original transformado en un texto completamente en mayúsculas (uppercase), en minúsculas (lowercase) o con la primera letra de cada palabra en mayúscula (capitalize).

Veamos un ejemplo con ambas: **Ej06.02b-DecoTransform.html:**

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Decoration / Transform</title>
   <style>
      #sp1 {   text-decoration: underline;          text-transform: lowercase; }
      #sp2 {   text-decoration: line-through;       }
      #sp3 {   text-decoration: overline;           text-transform: uppercase;}
      #sp4 {   text-decoration: blink;              text-transform: capitalize; }
   </style>
</head>

<body>
      <section>
      <p>La <span id="sp1">PROPIEDAD</span> que
      <span id="sp2">modifica</span> decora el
      <span id="sp3">texto</span> se denomina
      <span id="sp4">text-decoration.</span></p>
      </section>
</body>
</html>
```

Uno de los principales problemas del diseño de documentos y páginas mediante CSS consiste en la alineación vertical en una misma línea de varios elementos diferentes como imágenes y texto. Para controlar esta alineación, CSS define la propiedad vertical-align.

Propiedad	**vertical-align**
Valores	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <porcentaje> | <medida> | inherit
Se aplica a	Elementos en línea y celdas de tabla
Valor inicial	baseline
Descripción	Determina la alineación vertical de los contenidos de un elemento

El valor por defecto es baseline y el valor más utilizado cuando se establece la propiedad vertical-align es middle. Veamos un ejemplo:

Ej06.02c-VerticalAlign.html

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

<head>
   <meta charset="UTF-8" />
   <title>VerticalAlign </title>
   <style>
      #contenedor {
         width: 80%;
         margin: 0 auto;
      }

      .caja {  width: 30%;
               float: left;
               height: 140px;
               margin: 10px;
               font-size: 25px;
               border: 1px solid grey;       }

      img {    margin-top: 20px;
               border: 1px dashed red;
               width: 80px;                  }

      p  {     float: right;          margin-right: 10px;
               font-weight: bold;            }

   </style>
</head>
<body>
   <section>
      <div id="contenedor">
         <div class="caja">   <img src="../img/help.png" alt="Ayuda" />
             <span style="vertical-align: baseline">Ayuda</span>   <p>Baseline</p>
          </div>

          <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
             <span style="vertical-align: sub">Ayuda</span>         <p>Sub</p>
          </div>

          <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
             <span style="vertical-align: super">Ayuda</span>      <p>Super</p>
          </div>

          <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
             <span style="vertical-align: top">Ayuda</span>               <p>Top</p>
          </div>

          <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
             <span style="vertical-align: text-top">Ayuda</span>   <p>Text-Top</p>
          </div>
```

```
        <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
           <span style="vertical-align: middle">Ayuda</span>     <p>Middle</p>
        </div>

        <div class="caja">  <img  id="esp" src="../img/help.png" alt="Ayuda" />
           <span style="vertical-align: bottom">Ayuda</span>     <p>Bottom</p>
        </div>

        <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
           <span style="vertical-align: text-bottom">Ayuda</span> <p>Text Bottom</p>
        </div>

        <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
           <span style="vertical-align: 1em">Ayuda</span>               <p>Medida 1em</p>
        </div>

        <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
           <span style="vertical-align: -1em">Ayuda</span>       <p>Medida -1em</p>
        </div>

        <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
           <span style="vertical-align: 10%">Ayuda</span>               <p>Porc. 10%</p>
        </div>

        <div class="caja">  <img src="../img/help.png" alt="Ayuda" />
           <span style="vertical-align: 80%">Ayuda</span>               <p>Porc. 80%</p>
        </div>
     </div>
   </section>
</body>
</html>
```

En muchas publicaciones impresas suele ser habitual tabular la primera línea de cada párrafo para facilitar su lectura. CSS permite controlar esta tabulación mediante la propiedad text-indent.

Propiedad	**text-indent**
Valores	<medida> | <porcentaje> | inherit
Se aplica a	Los elementos de bloque y las celdas de tabla
Valor inicial	0
Descripción	Tabula desde la izquierda la primera línea del texto original

CSS también permite controlar la separación entre las letras que forman las palabras y la separación entre las palabras que forman los textos. La propiedad que controla la separación entre letras se llama letter-spacing y la separación entre palabras se controla mediante word-spacing.

Propiedad	**letter-spacing**
Valores	normal | <medida> | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las letras que forman las palabras del texto

Propiedad	**word-spacing**
Valores	normal | <medida> | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las palabras que forman el texto

Veamos un ejemplo con las últimas 3 propiedades: **Ej06.02d-Varios.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>Propiedades Texto</title>
   <style>
      div { width: 250px;     height: 250px;
            border: 2px solid blue;}
      p { text-indent: -3em;  letter-spacing: -2px;
         word-spacing: 10px;  padding: 40px;}
   </style>
</head>
<body>
   <section>
      <div>
         <p>En muchas publicaciones impresas suele ser habitual tabular la primera línea de
         cada párrafo para facilitar su lectura.
         CSS permite controlar esta tabulación mediante la propiedad text-indent.</p>
      </div>
   </section>
</body></html>
```

Cuando se utiliza un valor numérico en las propiedades letter-spacing y word-spacing, se interpreta como la separación adicional que se añade (si el valor es positivo) o se quita (si el valor es negativo) a la separación por defecto entre letras y palabras respectivamente.

Como ya se sabe, el tratamiento que hace HTML de los espacios en blanco es uno de los aspectos más difíciles de comprender cuando se empiezan a crear las primeras páginas web. Básicamente, HTML elimina todos los espacios en blanco sobrantes, es decir, todos salvo un espacio en blanco entre cada palabra.

Para forzar los espacios en blanco adicionales se debe utilizar la entidad HTML   y para forzar nuevas líneas, se utiliza el elemento <br/>. Además, HTML proporciona el elemento <pre> que muestra el contenido tal y como se escribe, respetando todos los espacios en blanco y todas las nuevas líneas.

CSS también permite controlar el tratamiento de los espacios en blanco de los textos mediante la propiedad white-space.

Propiedad	**white-space**
Valores	normal | pre | nowrap | pre-wrap | pre-line | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el tratamiento de los espacios en blanco del texto

El significado de cada uno de los valores es el siguiente:

- **normal:** comportamiento por defecto de HTML.
- **pre:** se respetan los espacios en blanco y las nuevas líneas (exactamente igual que la etiqueta <pre>). Si la línea es muy larga, se sale del espacio asignado para ese contenido.
- **nowrap:** elimina los espacios en blanco y las nuevas líneas. Si la línea es muy larga, se sale del espacio asignado para ese contenido.
- **pre-wrap:** se respetan los espacios en blanco y las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.
- **pre-line:** elimina los espacios en blanco y respeta las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.

En la siguiente tabla se resumen las características de cada valor:

Valor	Respeta espacios en blanco	Respeta saltos de línea	Ajusta las líneas
normal	no	no	si
pre	si	si	no
nowrap	no	no	no
pre-wrap	si	si	si
pre-line	no	si	si

Ej06.02e-WhiteSpace.html

```
<!DOCTYPE html><html lang="es">
<head>
   <meta charset="UTF-8" />
   <title>WhiteSpace</title>
   <style>
      div {width: 50%; border: medium solid #000; padding: .5em; margin: 1em 0;}
      #p1 {white-space: normal; margin: 0;}
      #p2 {white-space: pre; margin: 0;}
      #p3 {white-space: pre-wrap; margin: 0;}
      #p4 {white-space: nowrap; margin: 0;}
      #p5 {white-space: pre-line; margin: 0;}
   </style>
</head>
<body>
      <div>
<p  id="p1"><strong>[white-space: normal]</strong> CSS también permite controlar      el
tratamiento de los
   espacios en blanco               de los textos mediante
   la propiedad white-space.</p>
</div>
<div>
<p id="p2"><strong>[white-space: pre]</strong> CSS también permite controlar      el
tratamiento de los
   espacios en blanco               de los textos mediante
   la propiedad white-space.</p>
</div>
<div>
<p id="p3"><strong>[white-space: pre-wrap]</strong> CSS también permite controlar      el
tratamiento de los
   espacios en blanco               de los textos mediante
   la propiedad white-space.</p>
</div>
<div>
<p id="p4"><strong>[white-space: nowrap]</strong> CSS también permite controlar      el
tratamiento de los
   espacios en blanco               de los textos mediante
   la propiedad white-space.</p>
</div>

<div>
<p id="p5"><strong>[white-space: pre-line]</strong> CSS también permite controlar      el
tratamiento de los
   espacios en blanco               de los textos mediante
   la propiedad white-space.</p>
</div>
</body>
</html>
```

Por último, CSS define unos elementos especiales llamados "pseudo-elementos" que permiten aplicar estilos a ciertas partes de un texto. En concreto, CSS permite definir estilos especiales a la primera línea y a la primera letra de un texto.

El pseudo-elemento **:first-line** permite aplicar estilos a la primera línea de un texto. Las palabras que forman la primera línea de un texto dependen del espacio reservado para mostrar el texto o del tamaño de la ventana del navegador, por lo que CSS calcula de forma automática las palabras que forman la primera línea de texto en cada momento.

De la misma forma, CSS permite aplicar estilos a la primera letra del texto mediante el pseudo-elemento **:first-letter**. La siguiente imagen muestra el uso del pseudo-elemento :first-letter para crear una letra capital:

Veamos un ejemplo con ambos pseudo-elementos:

Ej06.02f-Pseudoelementos.html

```
<!DOCTYPE html>
<html lang="es">  <!-- Ej06.02f-Pseudoelementos.html-->
<head>
   <meta charset="UTF-8" />
   <title>Pseudoelementos</title>
   <style>
      p:first-line { font-weight: bold;
                  line-height: 25px;}
      p:first-letter {
         font-size: 50px;
         color: lime;
         vertical-align: -10px;
         text-shadow: 5px 5px 5px grey;
      }

      p {text-indent: -50px;
         padding: 50px;}
   </style>
</head>
<body>
   <section>
      <div>
         <h3>La riojana JMP desarrollará el control del primer reactor de fusión del
mundo</h3>
         <p>El primer reactor nuclear de fusión (ITER) que se construye en el mundo  para
         instalarse en la central nuclear por fusión que se edifica en Cadarache (Francia)
         tendrá una profunda huella riojana. De hecho, el consorcio internacional evaluador
         de la UE, tras elegir como el mejor el proyecto presentado por JMP Ingenieros,
         ha encargado a la compañía ubicada en Sotés desde 1992 el desarrollo del sistema
         de instrumentación y control electrónico del reactor nuclear de fusión.
         </p>
      </div>
   </section>
</body>
</html>
```

Ejercicio7

A partir del HTML y del CSS del Ejercicio 5 RESUELTO (ver página 69) se pide:

- La fuente base de la página debe ser: color negro, tipo Arial, tamaño 0.9em, interlineado 1.4em.
- Los elementos <h3> de .articulo se muestran en color #CC6600, con un tamaño de letra de 1.6em, un interlineado de 1.2em y un margen inferior de 0.3em.
- Los elementos del #menu deben mostrar un margen de 1em y los enlaces deben ser de color blanco y tamaño de letra 1.3em.
- El tamaño del texto de todos los contenidos de #lateral debe ser de 1em. La fecha de cada noticia debe ocupar el espacio de toda su línea y mostrarse en color gris claro #999. El elemento <h3> de #noticias debe mostrarse de color #003366.
- El texto del elemento #publicidad es de color gris oscuro #555 y todos los enlaces de color #CC6600.
- Los enlaces contenidos dentro de .articulo son de color #CC6600 y todos los párrafos muestran un margen superior e inferior de 0.5em.
- Añadir las reglas necesarias para que el contenido de #secundario y de #pie se vean como en la imagen que se muestra.
  Una pista: Seguir un patrón similar a los titulares de los artículos.

El resultado será:

Para acabar el tema, un menú desplegable multinivel **Ej06.00-MenuDesplegable.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <style type="text/css">
   /* Con esto centramos el menú*/
   div {       margin: 0 auto;       width: 300px;  }

   /* A TODOS los elementos le quitamos los puntos y las rayas de los enlaces */
   *   {    font-family: Verdana;
            list-style:none;
            text-decoration:none;
            margin:0;   padding:0;    }

   /* Los list DESCENDIENTES directos del menú principal, se muestran en Vertical*/
   .menu > li {      float:left;      }

   /* Para los elementos del menú principal (que son enlaces)*/
   .menu li a {      background:#0c9ba0;
                     color:#FFF;
                     display:block;
                     border:2px solid white;
                     padding:10px 12px;      }

   /* Al pasar el ratón por encima se pone mas clarito*/
   .menu li a:hover {         background:#0fbfc6;    }

   /*Por defecto todos los submenús no serán visibles, los ocultaremos usando display:
   none. Todos los submenús tendrán un ancho mínimo de 140px para que no se vean desiguales
   y llevaran  position: absolute para que no afecten el ancho del menu principal.*/
   .menu li ul {     display:none;
                     position:absolute;
                     min-width:140px;                }

   /* Cuando un elemento tenga un subelemento, al pasar por encima se mostrará*/
   .menu li:hover > ul {      display:block;                }

   /* Con estas 2 últimas reglas, los submenús de los submenús (y en adelante)
   se mostrarán a la derecha.*/
   .menu li ul li {     position:relative;   }
   .menu li ul li ul {  left:140px;   top:0;  }
  </style> </head>
<body>
   <div> <ul class="menu">
            <li><a href="">Módulos</a>
               <ul>
                  <li><a href="">Diseño Web</a>
                     <ul>
                        <li><a href="">HTML 4/5</a></li>
                        <li><a href="">CSS 2/3</a></li>
                        <li><a href="">CMS</a>
                              <ul>
                                     <li><a href="">Joomla</a></li>
                                     <li><a href="">Drupal</a></li>
                                     <li><a href="">Wordpress</a></li>
                              </ul>
                        </li>
                     </ul>
                  </li>
                  <li><a href="">Aplicaciones</a>
                     <ul>
                        <li><a href="">C++</a></li>
                        <li><a href="">Java6</a></li>
                        <li><a href="">Android 2</a></li>
                     </ul>
                  </li>
               </ul>
            </li>
         </ul>    </div>
</body> </html>
```

7 ENLACES

7.1 Estilos básicos

7.1.1 Tamaño, color y decoración

Los estilos más sencillos que se pueden aplicar a los enlaces son los que modifican su tamaño de letra, su color y la decoración del texto del enlace. Utilizando las propiedades text-decoration y font-weight se pueden conseguir múltiples estilos. Un ejemplo: **Ej07.01a-Enlaces.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Enlaces</title>
  <style type="text/css">
   a { margin: 1em 0; display: block;}

    .alternativo {color: #CC0000;}
    .simple {text-decoration: none;}
    .importante {font-weight: bold; font-size: 1.3em;}
    .especial {text-decoration:overline;}
  </style>
</head>
<body>
    <section>
      <a href="#">Enlace con el estilo por defecto</a>
      <a class="alternativo" href="#">Enlace de color rojo</a>
      <a class="simple" href="#">Enlace sin subrayado</a>
      <a class="importante" href="#">Enlace muy importante</a>
      <a class="especial" href="#">Enlace con un estilo especial</a>
    </section>
</body>
</html>
```

7.1.2 Pseudo-clases

CSS también permite aplicar diferentes estilos a un mismo enlace en función de su estado. De esta forma, es posible cambiar el aspecto de un enlace cuando por ejemplo el usuario pasa el ratón por encima o cuando el usuario pincha sobre ese enlace.

Como con los atributos id o class no es posible aplicar diferentes estilos a un mismo elemento en función de su estado, CSS introduce un nuevo concepto llamado pseudo-clases. En concreto, CSS define las siguientes cuatro pseudo-clases:

- **:link,** aplica estilos a los enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- **:visited,** aplica estilos a los enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario. El historial de enlaces visitados se borra automáticamente cada cierto tiempo y el usuario también puede borrarlo manualmente.
- **:hover,** aplica estilos al enlace sobre el que el usuario ha posicionado el puntero del ratón.
- **:active,** aplica estilos al enlace que está pinchando el usuario. Los estilos sólo se aplican desde que el usuario pincha el botón del ratón hasta que lo suelta, por lo que suelen ser unas pocas décimas de segundo.

Las pseudo-clases siempre incluyen dos puntos (:) por delante de su nombre y siempre se añaden a continuación del elemento al que se aplican, sin dejar ningún espacio en blanco por delante:
a: visited { ... } /* Incorrecto: la pseudo-clase no se puede separar de los dos puntos iniciales */
a :visited { ... } /* Incorrecto: la pseudo-clase siempre se añade sin espacios tras el elemento */
a:visited { ... } /* Correcto: la pseudo-clase modifica el comportamiento del elemento <a> */

Veamos un ejemplo curioso con transiciones de CSS3: **Ej07.01b-EnlaceChulo.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Enlace Chulo</title>
  <style type="text/css">
    a { display: block;   color: white;
    text-decoration: none;    }
    a:hover { color: red;}

    .enlaceguay {     width:100px;    height:50px;
                      font-size:16px; margin: 0 auto;
                      background:#101921;
                      text-align: center;
                      border: 6px solid #89A;
                      border-radius: 10px;
                      padding-top: 20px;

                      /* Transiciones CSS*/
                      /*-webkit-transition-property:width,height; -> Para Chrome*/
                      transition-property:width,height;
                      transition-duration:2s;
                      }

  .enlaceguay:hover { width:300px;    height:200px;
                      font-size:64px;
                      background:#606971;
                      border-radius: 150px;
                      padding-top: 100px;
                      transition-property:width,height;
                      transition-duration:2s;  }
  </style>
</head>
<body>
    <section>
      <div class="enlaceguay"><a href="#">¿Queréis mas?</a></div>
    </section>
</body>
</html>
```

Las pseudo-clases también se pueden combinar con cualquier otro selector complejo:

```
a.importante:visited { ... }
a#principal:hover { ... }
div#menu ul li a.primero:active { ... }
```

Cuando se aplican varias pseudo-clases diferentes sobre un mismo enlace, se producen colisiones entre los estilos de algunas pseudo-clases. Si se pasa por ejemplo el ratón por encima de un enlace visitado, se aplican los estilos de las pseudo-clases :hover y :visited. Si el usuario pincha sobre un enlace no visitado, se aplican las pseudo-clases :hover, :link y :active y así sucesivamente.

Si se definen varias pseudo-clases sobre un mismo enlace, el único orden que asegura que todos los estilos de las pseudo-clases se aplican de forma coherente es el siguiente: :link, :visited, :hover y :active. De hecho, en muchas hojas de estilos es habitual establecer los estilos de los enlaces de la siguiente forma:

```
a:link, a:visited {
  ...
}

a:hover, a:active {
  ...
}
```

Las pseudo-clases :link y :visited solamente están definidas para los enlaces, pero las pseudo-clases :hover y :active se definen para todos los elementos HTML. Desafortunadamente, Internet Explorer 6 y sus versiones anteriores solamente las soportan para los enlaces.

También es posible combinar en un mismo elemento las pseudo-clases que son compatibles entre sí:

```
/* Los estilos se aplican cuando el usuario pasa el ratón por encima de un
   enlace que todavía no ha visitado */
a:link:hover { ... }

/* Los estilos se aplican cuando el usuario pasa el ratón por
encima de un  enlace que ha visitado previamente */
a:visited:hover { ... }
```

Enlace número 1

Enlace número 2

Enlace número 3

Enlace número 4

Enlace número 5

Ejercicio08

Definir las reglas CSS que permiten mostrar los enlaces con los siguientes estilos:

1. En su estado normal, los enlaces se muestran de color rojo #CC0000.
2. Cuando el usuario pasa su ratón sobre el enlace, se muestra con un color de fondo rojo #CC0000 y la letra de color blanco #FFF.
3. Los enlaces visitados se muestran en color gris claro #CCC.

El resultado será muy similar a la imagen de la derecha.

7.2 Estilos avanzados

7.2.1 Decoración personalizada

Enlace con el estilo por defecto

Enlace un subrayado de otro color

Enlace con un subrayado muy ancho

Enlace con un subrayado muy separado

Enlace con un subrayado discontinuo

La propiedad text-decoration permite añadir o eliminar el subrayado de los enlaces. Sin embargo, el aspecto del subrayado lo controla automáticamente el navegador, por lo que su color siempre es el mismo que el del texto del enlace y su anchura es proporcional al tamaño de letra.

No obstante, utilizando la propiedad border-bottom es posible añadir cualquier tipo de subrayado para los enlaces de las páginas. El siguiente ejemplo muestra algunos enlaces con el subrayado personalizado: **Ej07.02a-EnlacesAv.html**

```
<!DOCTYPE html>
<html lang="es">
 <head> <meta charset="UTF-8" />
 <title>Enlaces Avanzados</title>
 <style type="text/css">
  a { margin: 1em 0; float: left; clear: left; text-decoration: none;}
  .simple {text-decoration: underline;}
  .color { border-bottom: medium solid #CC0000;}
  .ancho {border-bottom: thick solid;}
  .separado {border-bottom: 1px solid; padding-bottom: .6em;}
  .discontinuo {border-bottom: thin dashed;}
 </style>
</head>
<body>
      <a class="simple" href="#">Enlace con el estilo por defecto</a>
      <a class="color" href="#">Enlace un subrayado de otro color</a>
      <a class="ancho" href="#">Enlace con un subrayado muy ancho</a>
      <a class="separado" href="#">Enlace con un subrayado muy separado</a>
      <a class="discontinuo" href="#">Enlace con un subrayado discontinuo</a>
</body>
</html>
```

7.2.2 Imágenes según el tipo de enlace

En ocasiones, puede resultar útil incluir un pequeño icono al lado de un enlace para indicar el tipo de contenido que enlaza, como por ejemplo un archivo comprimido o un documento PDF.

Este tipo de imágenes son puramente decorativas en vez de imágenes de contenido, por lo que se deberían añadir con CSS y no con elementos de tipo <img>. Utilizando la propiedad background (y background-image) se puede personalizar el aspecto de los enlaces para que incluyan un pequeño icono a su lado.

La técnica consiste en mostrar una imagen de fondo sin repetición en el enlace y añadir el padding necesario al texto del enlace para que no se solape con la imagen de fondo.
Para empezar, veamos los iconos, de 16px y en formato png: **Ej07.02b-EnlacesImagen.html**
http://www.iconfinder.com/search/?q=pdf y http://www.iconfinder.com/search/?q=rss

Enlace a un archivo RSS

Enlace a un documento PDF

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Enlaces con Imagen</title>
  <style type="text/css">
    a { margin: 1em 0; float: left; clear: left; }

    .rss {  color: #E37529;   padding: 0 0 0 18px;
      background: #FFF
      url("http://cdn3.iconfinder.com/data/icons/socialnetworking/16/rss.png")
      no-repeat left center;
    }

    .pdf {     padding: 0 0 0 22px;
               background: #FFF
      url("http://cdn1.iconfinder.com/data/icons/silk2/page_white_acrobat.png")
               no-repeat left center;
    }
  </style>
</head>
<body>
      <a href="#">Enlace con el estilo por defecto</a>
      <a class="rss" href="#">Enlace a un archivo RSS</a>
      <a class="pdf" href="#">Enlace a un documento PDF</a>
</body>
</html>
```

7.2.3 Mostrar los enlaces como si fueran botones

Las propiedades definidas por CSS permiten mostrar los enlaces con el aspecto de un botón, lo que puede ser útil en formularios en los que no se utilizan elementos específicos para crear botones.**Ej07.02c-EnlacesBoton.html**

Guardar

Enviar

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Enlaces Botón</title>
  <style type="text/css">
   a {         margin: 1em 0; float: left; clear: left; }
   a.boton {   text-decoration: none;  color: #222;
               background: #EEE;       border: 1px outset #CCC;
               padding: .1em .5em;           }
   a.boton:hover {   background: #CCB;              }
   a.boton:active {  border: 1px inset #000;        }
  </style>
</head>
<body>
      <a class="boton" href="#">Guardar</a>
      <a class="boton" href="#">Enviar</a>
    </section>

</body>
</html>
```

8 IMÁGENES

8.1 Estilos básicos

8.1.1 Establecer la anchura y altura de las imágenes

Utilizando las propiedades width y height, es posible mostrar las imágenes con cualquier altura/anchura, independientemente de su altura/anchura real:

```
#destacada {
  width: 120px;
  height: 250px;
}

<img id="destacada" src="imagen.png" />
```

No obstante, si se utilizan alturas/anchuras diferentes de las reales, el navegador deforma las imágenes y el resultado estético es muy desagradable.

Por otra parte, establecer la altura/anchura de todas las imágenes mediante CSS es una práctica poco recomendable. El motivo es que normalmente dos imágenes diferentes no comparten la misma altura/anchura. Por lo tanto, se debe crear una nueva regla CSS (y un nuevo selector) para cada una de las diferentes imágenes del sitio web.

En la práctica, esta forma de trabajo produce una sobrecarga de estilos que la hace inviable. Por este motivo, aunque es una solución que no respeta la separación completa entre contenidos (XHTML) y presentación (CSS), se recomienda establecer la altura/anchura de las imágenes mediante los atributos width y height de la etiqueta <img>:

```
<img src="imagen.png" width="120" height="250" />
```

De todos modos, la mejor solución es optar por usar CSS pero indicando, únicamente, una de las dos propiedades para el dimensionado, width o height.
Para el ejemplo usaremos http://recursostic.educacion.es/bancoimagenes/web/

Y buscaremos moulin rouge. Ahora el ejemplo, **Ej08.01a-TamImagen.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Enlaces Botón</title>
  <style type="text/css">
    img { width: 100px;
          /*height: 100px;*/}
  </style>
</head>
<body>
    <section>
      <img src="../img/Moulin_Rouge.jpg" alt="Moulin Rouge" />
    </section>
</body>
</html>
```

8.1.2 Eliminar el borde de las imágenes con enlaces

Cuando una imagen forma parte de un enlace, los navegadores muestran por defecto un borde azul grueso alrededor de las imágenes. Por tanto, una de las reglas más utilizadas en los archivos CSS es la que elimina los bordes de las imágenes con enlaces:

```
img {
  border: none;
}
```

Ejercicio 9

Definir las reglas CSS que permiten mostrar una galería de imágenes similar a la que se muestra en la siguiente imagen:

8.2 Estilos avanzados

8.2.1 Sombra (drop shadow)

La mayoría de aplicaciones de diseño gráfico permiten añadir una sombra (llamada drop shadow en inglés) a las imágenes. CSS no incluye propiedades que permitan mostrar de forma sencilla sombras en los elementos.

No obstante, existen varias técnicas sencillas y otras más avanzadas que permiten crear sombras que se adapten a cualquier imagen o elemento. A continuación se muestra una técnica sencilla para añadir sombra a las imágenes. **Ej08.02a-DropShadow.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Drop Shadow</title>
  <style type="text/css">
  .sombra {    float:left;    clear:left;
    background: url("http://wubbleyew.com/tests/images/shadowAlpha.png")
    no-repeat bottom right !important;
    background: url("http://wubbleyew.com/tests/images/shadow.gif)")
    no-repeat bottom right;
                    margin: 10px 0 10px 10px !important;
                    margin: 10px 0 10px 5px;
                    padding: 0px;    }

  .caja {           position:relative;      bottom:6px;
                    right: 6px;             padding:4px;
                    border: 1px solid #999999;
                    margin: 0px 0px 0px 0px;   }
  </style>
</head>
<body>
  <section>
    <div class="sombra">
      <div class="caja">
        <img src="../img/moulin_rouge.jpg" alt="Imagen genérica" />
      </div>
    </div>

    <div class="sombra">
      <div class="caja">
        <h4>Párrafo de texto</h4>
        <p>Esta técnica no sólo permite añadir sombra a las imágenes,
        sino a cualquier elemento.</p>
      </div>
    </div>
  </section>
</body>
</html>
```

La técnica mostrada se presentó por primera vez en la página web http://wubbleyew.com/tests/dropshadows.htm y consiste en la utilización de un par de elementos alrededor del elemento que va a mostrar una sombra. La sombra se consigue mediante una imagen muy grande que contiene una sombra orientada hacia el lado derecho e inferior.

La ventaja de este método es que es sencillo y solamente requiere añadir un par de por cada elemento. Además, como la sombra es una imagen muy grande, el efecto funciona con elementos de cualquier tamaño.

El mayor inconveniente de este método es que la sombra siempre se muestra en la misma dirección (derecha inferior) y que en Internet Explorer 6 y versiones anteriores sólo muestran la sombra correctamente cuando el color de fondo de la página es blanco (ya que Internet Explorer 6 y versiones anteriores no soportan imágenes en formato PNG con transparencias).

9 LISTAS

9.1 Estilos básicos

9.1.1 Viñetas personalizadas

Por defecto, los navegadores muestran los elementos de las listas no ordenadas con una viñeta formada por un pequeño círculo de color negro. Los elementos de las listas ordenadas se muestran por defecto con la numeración decimal utilizada en la mayoría de países.

No obstante, CSS define varias propiedades para controlar el tipo de viñeta que muestran las listas, además de poder controlar la posición de la propia viñeta. La propiedad básica es la que controla el tipo de viñeta que se muestra y que se denomina list-style-type.

Propiedad	**list-style-type**
Valores	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none | inherit
Se aplica a	Elementos de una lista
Valor inicial	disc
Descripción	Permite establecer el tipo de viñeta mostrada para una lista

En primer lugar, el valor none permite mostrar una lista en la que sus elementos no contienen viñetas, números o letras. Se trata de un valor muy utilizado, ya que es imprescindible para los menús de navegación creados con listas, como se verá más adelante.

El resto de valores de la propiedad list-style-type se dividen en tres tipos: gráficos, numéricos y alfabéticos.

- Los valores gráficos son disc, circle y square y muestran como viñeta un círculo relleno, un círculo vacío y un cuadrado relleno respectivamente.
- Los valores numéricos están formados por decimal, decimal-leading-zero, lower-roman, upper-roman, armenian y georgian.
- Por último, los valores alfanuméricos se controlan mediante lower-latin, lower-alpha, upper-latin, upper-alpha y lower-greek.

Veamos un ejemplo **Ej09.01a-ListasPropias.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Listas sPropias</title>
  <style type="text/css">
    ul  { list-style: circle;   }
    ol  { list-style: katakana; }
  </style>
</head>
<body>
  <section>
    <ul>
      <li>Elemento desordenado 1</li>
      <li>Elemento desordenado 2</li>
      <li>Elemento desordenado 3</li>
    </ul>
    <ol>
      <li>Elemento Ordenado 1</li>
      <li>Elemento Ordenado 2</li>
      <li>Elemento Ordenado 3</li>
    </ol>
  </section>
</body>
</html>
```

La propiedad list-style-position permite controlar la colocación de las viñetas.

Propiedad	**list-style-position**
Valores	inside | outside | inherit
Se aplica a	Elementos de una lista
Valor inicial	outside
Descripción	Permite establecer la posición de la viñeta de cada elemento de una lista

Mediante Outside, el texto de la segunda línea aparecerá debajo del inicio del texto de la primera, frente al valor Inside que haré que el texto de la segunda línea aparezca justo debajo de la viñeta. El valor predeterminado es Outside. Un ejemplo. **Ej09.01b-ListaOutsideInside.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Listas Outside / Inside</title>
  <style type="text/css">
    div { margin 0 auto;     width: 200px;}
    ul  { list-style-position: inside; }
  </style>
</head>
<body>
    <div>
      <ul>
        <li>Mediante Outside, el texto de la segunda línea aparecerá debajo del inicio del
         texto de la primera, frente al valor Inside que haré que el texto de la segunda
         línea aparezca justo debajo de la viñeta. Lo vemos con un ejemplo.</li>
      </ul>
    </div>
</body></html>
```

Utilizando las propiedades anteriores (list-style-type y list-style-position), se puede seleccionar el tipo de viñeta y su posición, pero no es posible personalizar algunas de sus características básicas como su color y tamaño. Cuando se requiere personalizar el aspecto de las viñetas, se debe emplear la propiedad list-style-image, que permite mostrar una imagen propia en vez de una viñeta automática.

Propiedad	**list-style-image**
Valores	<url> | none | inherit
Se aplica a	Elementos de una lista
Valor inicial	none
Descripción	Permite reemplazar las viñetas automáticas por una imagen personalizada

Las imágenes personalizadas se indican mediante la URL de la imagen. Si no se encuentra la imagen o no se puede cargar, se muestra la viñeta automática correspondiente (salvo que explícitamente se haya eliminado mediante la propiedad list-style-type).
Un ejemplo. **Ej09.01c-ListaImagen.html**

```
<!DOCTYPE html> <html lang="es">
  <head> <meta charset="UTF-8" />
  <title>Imagen en Listas</title>
  <style type="text/css">
    ul  { list-style-image:
    url("http://cdn3.iconfinder.com/data/icons/socialnetworking/16/rss.png"); }
  </style>
</head>
<body>
    <div>
      <ul>
        <li>Elemento1</li>    <li>Elemento2</li>    <li>Elemento3</li>
      </ul>
    </div>
</body></html>
```

Como es habitual, CSS define una propiedad de tipo "shorthand" que permite establecer todas las propiedades de una lista de forma directa. La propiedad se denomina list-style.

Propiedad	**list-style**
Valores	(<list-style-type> || <list-style-position> || <list-style-image>) | inherit
Se aplica a	Elementos de una lista
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultánea todas las opciones de una lista

En la definición anterior, la notación || significa que el orden en el que se indican los valores de la propiedad es indiferente. El siguiente ejemplo indica que no se debe mostrar ni viñetas automáticas ni viñetas personalizadas:

```
ul { list-style: none }
```

Cuando se utiliza una viñeta personalizada, es conveniente indicar la viñeta automática que se mostrará cuando no se pueda cargar la imagen:

```
ul { list-style: url("imagenes/cuadrado_rojo.gif") square; }
```

9.1.2 Menú vertical

Los sitios web correctamente diseñados emplean las listas de elementos para crear todos sus menús de navegación. Utilizando la etiqueta <ul> de HTML se agrupan todas las opciones del menú y haciendo uso de CSS se modifica su aspecto para mostrar un menú horizontal o vertical.

A continuación se muestra la transformación de una lista sencilla de enlaces en un menú vertical de navegación. **Ej09.01d-MenuVertical.html.** Lista de enlaces original:

```
<ul class="menu">
  <li><a href="#">Elemento 1</a></li>
  <li><a href="#">Elemento 2</a></li>
  <li><a href="#">Elemento 3</a></li>
  <li><a href="#">Elemento 4</a></li>
  <li><a href="#">Elemento 5</a></li>
  <li><a href="#">Elemento 6</a></li>
</ul>
```

El proceso de transformación de la lista en un menú requiere de los siguientes pasos:
1) Definir la anchura del menú:

```
ul.menu { width: 180px; }
```

2) Eliminar las viñetas automáticas y todos los márgenes y espaciados aplicados por defecto:

```
ul.menu {
...
  list-style: none;
  margin: 0;
  padding: 0;

}
```

3) Añadir un borde al menú de navegación y establecer el color de fondo y los bordes de cada elemento del menú:

```
ul.menu {
...
  border: 1px solid #7C7C7C;
  border-bottom: none;

}
```

```
ul.menu li {
  background: #F4F4F4;
  border-bottom: 1px solid #7C7C7C;
  border-top: 1px solid #FFF;
}
```

4) Aplicar estilos a los enlaces: mostrarlos como un elemento de bloque para que ocupen todo el espacio de cada <li> del menú, añadir un espacio de relleno y modificar los colores y la decoración por defecto:

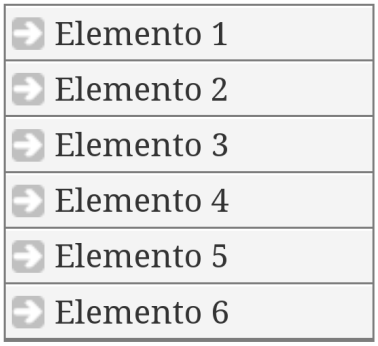
```
ul.menu li a {
  color: #333;
  display: block;
  padding: .2em 0 .2em .5em;
  text-decoration: none;
}
```

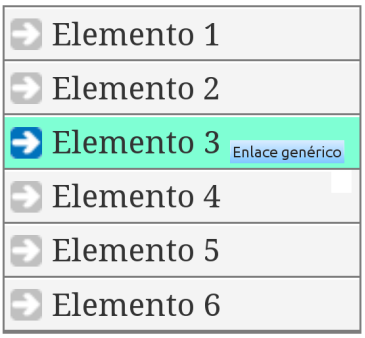
Los tipos de menús verticales que se pueden definir mediante las propiedades CSS son innumerables, como se puede observar en la página
http://www.exploding-boy.com/images/EBmenus/menus.html

Ejercicio10
Crear un menú vertical con las siguientes características:
1) Los elementos deben mostrar una imagen de fondo (flecha_inactiva.png):
2) Cuando se pasa el ratón por encima de un elemento, se debe mostrar una imagen alternativa (flecha_activa.png)
3) El color de fondo del elemento también debe variar ligeramente y mostrar un color gris más oscuro (#E4E4E4) cuando se pasa el ratón por encima
4) El comportamiento anterior se debe producir cuando el usuario pasa el ratón por encima de cualquier zona del elemento del menú, no solo cuando se pasa el ratón por encima del texto del elemento.

Enlaces para las flechas:
http://librosweb.es/ejercicios/css/comun/imagenes/flecha_inactiva.png
http://librosweb.es/ejercicios/css/comun/imagenes/flecha_activa.png

Forma en la que quedarán los menús:





9.2 Estilos avanzados

9.2.1 Menú horizontal básico

Partiendo de la misma lista de elementos del menú vertical, resulta muy sencillo crear un menú de navegación horizontal. La clave reside en modificar el posicionamiento original de los elementos de la lista. **Ej09.01e-MenuHorizontal.html.**

Código HTML del menú horizontal:

```
<ul class="menu">
  <li><a href="#">Elemento 1</a></li>
  <li><a href="#">Elemento 2</a></li>
  <li><a href="#">Elemento 3</a></li>
  <li><a href="#">Elemento 4</a></li>
  <li><a href="#">Elemento 5</a></li>
</ul>
```

El proceso de creación del menú horizontal consta de los siguientes pasos:
1) Aplicar los estilos CSS básicos para establecer el estilo del menú (similares a los del menú vertical anterior):

```
ul.menu {
  background: #F4F4F4;
  border: 1px solid #7C7C7C;
  list-style: none;
  margin: 0;
  padding: 0;
}

ul.menu li a {
  color: #333;
  display: block;
  padding: .3em;
  text-decoration: none;
}
```

2) Establecer la anchura de los elementos del menú. Como el menú es de anchura variable y contiene cinco elementos, se asigna una anchura del 20% a cada elemento. Si se quiere tener un control más preciso sobre el aspecto de cada elemento, es necesario asignar una anchura fija al menú.

Además, se posiciona de forma flotante los elementos de la lista mediante la propiedad float. Esta es la clave de la transformación de una lista en un menú horizontal:

```
ul.menu li {
  float: left;
  width: 20%;
}
```

Después de posicionar de forma flotante a todos los elementos de la lista, el elemento <ul> es un elemento vacío ya que en su interior no existe ningún elemento posicionado de forma normal.

Como ya se explicó en las secciones anteriores, la solución de este problema consiste en aplicar la propiedad overflow: hidden; al elemento <ul>, de forma que encierre a todos los elementos posicionados de forma flotante:

```
ul.menu {
  overflow: hidden;
}
```

3) Establecer los bordes de los elementos que forman el menú:

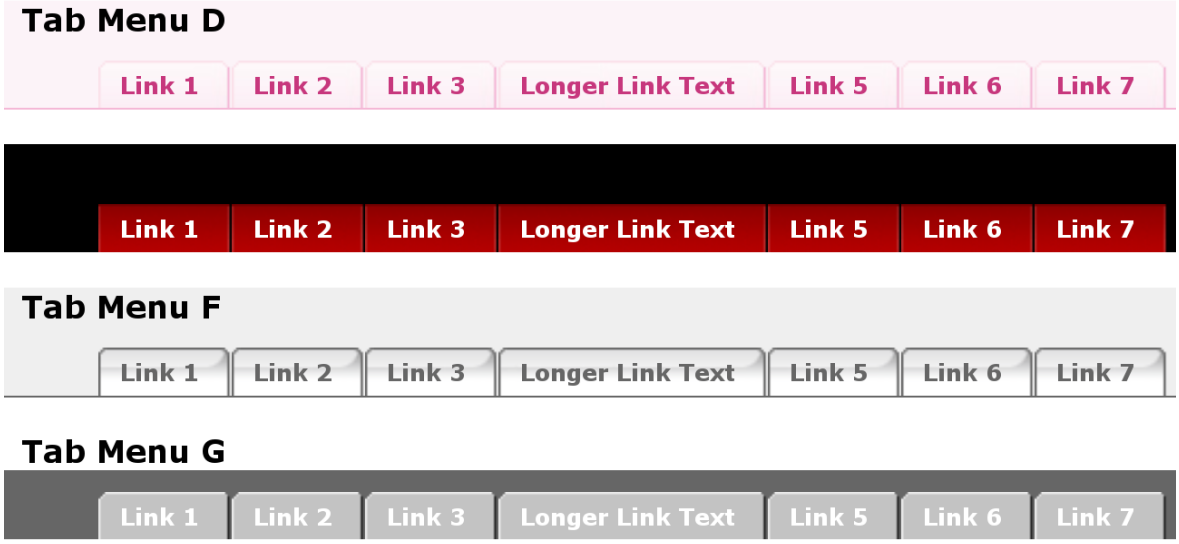
```
ul.menu li a {
  border-left: 1px solid #FFF;
  border-right: 1px solid #7C7C7C;
}
```

4) Por último, se elimina el borde derecho del último elemento de la lista, para evitar el borde duplicado:

```
<ul>
  <li><a href="#">Elemento 1</a></li>
  <li><a href="#">Elemento 2</a></li>
  <li><a href="#">Elemento 3</a></li>
  <li><a href="#">Elemento 4</a></li>
  <li><a href="#" style="border-right: none">Elemento 5</a></li>
</ul>
```

9.2.2 Menús Avanzados

Modificando los estilos de cada elemento del menú y utilizando imágenes de fondo y las pseudo-clases :hover y :active, se pueden crear menús horizontales complejos, incluso con el aspecto de un menú de solapas o pestañas:



El código fuente de los menús de la imagen anterior y muchos otros se puede encontrar en http://exploding-boy.com/images/cssmenus/menus.html

Además de los menús horizontales de pestañas, haciendo uso de las propiedades de CSS se pueden crear menús verticales y horizontales muy avanzados

Taming Lists // ALA

El código CSS de todos los ejemplos de la imagen anterior y muchos otros se pueden encontrar en: http://alvit.de/css-showcase/css-navigation-techniques-showcase.php

10 TABLAS

10.1 Estilos básicos

A	B	C	D	E
a	1	2	3	4
b	1	2	3	4
c	1	2	3	4
d	1	2	3	4

Cuando se aplican bordes a las celdas de una tabla, el aspecto por defecto con el que se muestra en un navegador es el siguiente de la imagen a la derecha:

El estándar CSS 2.1 define dos modelos diferentes para el tratamiento de los bordes de las celdas. La propiedad que permite seleccionar el modelo de bordes es border-collapse:

Propiedad	**border-collapse**
Valores	collapse | separate | inherit
Se aplica a	Todas las tablas
Valor inicial	separate
Descripción	Define el mecanismo de fusión de los bordes de las celdas adyacentes de una tabla

El modelo collapse fusiona de forma automática los bordes de las celdas adyacentes, mientras que el modelo separate fuerza a que cada celda muestre sus cuatro bordes. Por defecto, los navegadores utilizan el modelo separate, tal y como se puede comprobar en el ejemplo anterior.

En general, los diseñadores prefieren el modelo collapse porque estéticamente resulta más atractivo y más parecido a las tablas de datos tradicionales. El ejemplo anterior se puede rehacer para mostrar la tabla con bordes sencillos y sin separación entre celdas:

Ej10.01a-TablaCollapse.html

```
<!DOCTYPE html>
<html lang="es">
   <head>
   <meta charset="UTF-8" />
   <title>Tabla Collapase</title>
   <style type="text/css">
      /* con esta regla se fusionan las celdas
      tabla {
         border-collapse: collapse; }*/


      /* Estas 2 reglas equivalen a poner en HTML border="1"*/
      .normal {
         border: 1px outset grey;
      }

      .normal td {
         border: 1px inset grey;
      }
   </style>
</head>
<body>
  <table class="normal" summary="Tabla genérica">
    <tr>  <td>A</td>  <td>B</td>  <td>C</td>  <td>D</td>  <td>E</td>  </tr>
    <tr>  <td>a1</td> <td>b1</td> <td>c1</td> <td>d1</td> <td>e1</td> </tr>
    <tr>  <td>a2</td> <td>b2</td> <td>c2</td> <td>d2</td> <td>e2</td> </tr>
    <tr>  <td>a3</td> <td>b3</td> <td>c3</td> <td>d3</td> <td>e3</td> </tr>
    <tr>  <td>a4</td> <td>b4</td> <td>c4</td> <td>d4</td> <td>e4</td> </tr>
  </table>
</body>
</html>
```

Aunque parece sencillo, el mecanismo que utiliza el modelo collapse es muy complejo, ya que cuando los bordes que se fusionan no son exactamente iguales, debe tener en cuenta la anchura de cada borde, su estilo y el tipo de celda que contiene el borde (columna, fila, grupo de filas, grupo de columnas) para determinar la prioridad de cada borde.

Si se opta por el modelo separate (que es el que se aplica si no se indica lo contrario) se puede utilizar la propiedad border-spacing para controlar la separación entre los bordes de cada celda.

Propiedad	**border-spacing**
Valores	<medida> <medida>? | inherit
Se aplica a	Todas las tablas
Valor inicial	0
Descripción	Establece la separación entre los bordes de las celdas adyacentes de una tabla

Si solamente se indica como valor una medida, se asigna ese valor como separación horizontal y vertical. Si se indican dos medidas, la primera es la separación horizontal y la segunda es la separación vertical entre celdas. Veamos un ejemplo siguiendo el anterior:

Ej10.01b-TableSpacing.html

```
<!DOCTYPE html>
<html lang="es">
...
   .normal {
      width: 250px;
      border: 2px solid green;
      border-spacing: 2px;
   }
   .normal th, .normal td {
      border: 1px solid red;
  }
...
</body>
</html>
```

A	B	C	D	E
a	1	2	3	4
b	1	2	3	4
c	1	2	3	4
d	1	2	3	4

La propiedad border-spacing sólo controla la separación entre celdas y por tanto, no se puede utilizar para modificar el tipo de modelo de bordes que se utiliza. En concreto, si se establece un valor igual a 0 para la separación entre los bordes de las celdas, el resultado es muy diferente al modelo collapse.

En la tabla del ejemplo anterior, se ha establecido la propiedad border-spacing: 0, por lo que el navegador no introduce ninguna separación entre los bordes de las celdas. Además, como no se ha establecido de forma explícita ningún modelo de bordes, el navegador utiliza el modelo separate.

El resultado es que border-spacing: 0 muestra los bordes con una anchura doble, ya que en realidad se trata de dos bordes iguales sin separación entre ellos. Por tanto, las diferencias visuales con el modelo border-collapse: collapse son muy evidentes.

10.2 Estilos avanzados

CSS define otras propiedades específicas para el control del aspecto de las tablas. Una de ellas es el tratamiento que reciben las celdas vacías de una tabla, que se controla mediante la propiedad empty-cells. Esta propiedad sólo se aplica cuando el modelo de bordes de la tabla es de tipo separate.

Propiedad	**empty-cells**
Valores	show | hide | inherit
Se aplica a	Celdas de una tabla
Valor inicial	show
Descripción	Define el mecanismo utilizado para el tratamiento de las celdas vacías de una tabla

El valor hide indica que las celdas vacías no se deben mostrar. Una celda vacía es aquella que no tiene ningún contenido, ni siquiera un espacio en blanco o un .
Veamos un ejemplo: **Ej10.02a-EmptyCells.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>EmptyCells</title>
<style type="text/css">
  table  {  border-collapse:separate;
            empty-cells:hide;
            border: 1px solid grey;  }

  td     {   border: 1px solid;
            min-width: 100px;                }
</style>
</head>
<body>
  <table summary="Tabla">
    <tr>  <td>1</td>  <td></td>   <td>c1</td> </tr>
    <tr>  <td>4</td>  <td>5</td>  <td></td>   </tr>
    <tr>  <td></td>   <td>8</td>  <td>9</td>  </tr>
  </table>
</body>
</html>
```

El valor por defecto de la propiedad empty-cells es show, que hace que se muestren siempre todas las celdas. En el ejemplo anterior, podemos probar qué pasa al comentar la propiedad empty-cells.

Si todas las celdas de una misma fila están vacías y se indica la propiedad empty-cells: hide, la altura de la fila es cero y por tanto, en su lugar sólo se muestra uno de los dos border-spacing verticales de esa fila.

Por otra parte, el título de las tablas se establece mediante el elemento <caption>, que por defecto se muestra encima de los contenidos de la tabla. La propiedad caption-side permite controlar la posición del título de la tabla.

Propiedad	**caption-side**
Valores	top | bottom | inherit
Se aplica a	Los elementos caption
Valor inicial	top
Descripción	Establece la posición del título de la tabla

El valor bottom indica que el título de la tabla se debe mostrar debajo de los contenidos de la tabla. Si también se quiere modificar la alineación horizontal del título, debe utilizarse la propiedad text-align. Veamos un ejemplo modificando el anterior: **Ej10.02b-CaptionSide.html**

```
<!DOCTYPE html>
...
<title>CaptionSide</title>
...
  caption {
    caption-side: bottom;
    text-align: right;
  }
</style>
</head>
<body>
  <table summary="Tabla">
    <caption> Tabla de Prueba</caption>
...
</html>
```

Ejercicio 11

Determinar las reglas CSS necesarias para mostrar la siguiente tabla con el aspecto final mostrado en la imagen (modificar el código HTML que se considere necesario añadiendo los atributos class oportunos). La Tabla original será la siguiente:

Cambio	Compra	Venta	Máximo	Mínimo
Euro/Dolar	1.2524	1.2527	1.2539	1.2488
Dolar/Yen	119.01	119.05	119.82	119.82
Libra/Dolar	1.8606	1.8611	1.8651	1.8522
Euro/Yen	149.09	149.13	149.79	148.96

Los pasos serán los siguientes:

1) Alinear el texto de las celdas, cabeceras y título. Definir los bordes de la tabla, celdas y cabeceras (color gris oscuro #333).
2) Formatear las cabeceras de fila y columna con la imagen de fondo correspondiente en cada caso (fondo_gris.gif, euro.png, dolar.png, yen.png, libra.png). Modificar el tipo de letra de la tabla y utilizar Arial. El color azul claro es #E6F3FF.

Las imágenes a emplear son las siguientes:
http://librosweb.es/ejercicios/css/comun/imagenes/fondo_gris.gif
http://librosweb.es/ejercicios/css/comun/imagenes/euro.png
http://librosweb.es/ejercicios/css/comun/imagenes/dolar.png
http://librosweb.es/ejercicios/css/comun/imagenes/yen.png
http://librosweb.es/ejercicios/css/comun/imagenes/libra.png

3) Mostrar un color alterno en las filas de datos (color amarillo claro #FFFFCC).
4) Una vez acabado el apartado anterior, se podría completar añadiendo una pequeña mejora: que el color de las filas varíe cuando el usuario pasa el ratón por encima de cada fila con el color aguamarina y que, al pulsar, se convierta en rosa.

El resultado final del ejercicio será el siguiente:

Cambio	Compra	Venta	Máximo	Mínimo
€ Euro/Dolar	1.2524	1.2527	1.2539	1.2488
$ Dolar/Yen	119.01	119.05	119.82	119.82
£ Libra/Dolar	1.8606	1.8611	1.8651	1.8522
¥ Yen/Euro	0.6711	0.6705	0.6676	0.6713

11 FORMULARIOS

11.1 Estilos básicos

11.1.1 Mostrar un botón como un enlace

Como ya se vio anteriormente, el estilo por defecto de los enlaces se puede modificar para que se muestren como botones de formulario. Ahora, los botones de formulario también se pueden modificar para que parezcan enlaces.

Veamos un ejemplo: **Ej11.01a-BotonEnlace.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>Botón Enlace</title>
<style type="text/css">
  .enlace {
    border: 0;
    padding: 0;
    background-color: transparent;
    color: blue;
    border-bottom: 1px solid blue;
 }
</style>
</head>

<body>
<section>
<form action="maneja_formulario.php" method="post">
  <input type="submit" value="Botón normal" /> <br />
  <input class="enlace" type="submit" value="Botón como enlace" />
</form>
</section>
</body>
</html>
```

11.1.2 Mejoras en los campos de texto

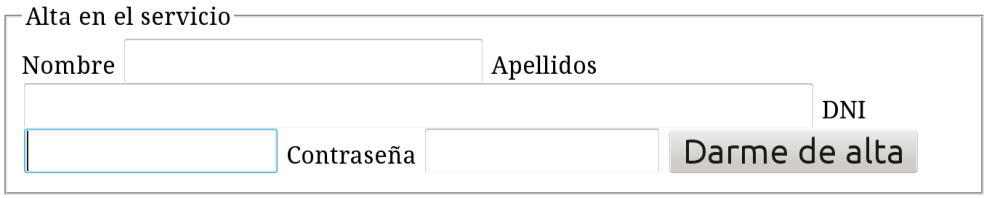
Por defecto, los campos de texto de los formularios no incluyen ningún espacio de relleno, por lo que el texto introducido por el usuario aparece pegado a los bordes del cuadro de texto.

Añadiendo un pequeño padding a cada elemento <input>, se mejora notablemente el aspecto del formulario. **Ej11.01b-MejoraInput.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>MejoraInput</title>
<style type="text/css">
  form input {
    padding: .5em;
    margin: 5px;
    border: 1px solid cyan;
  }
</style>
</head>

<body>
<section>
<form action="maneja_formulario.php" method="post">
  Usuario: <input type="text" name="usuario"/> <br />
  Contraseña: <input type="password" name="clave"/> <br />
</form>
</section>
</body>
</html>
```

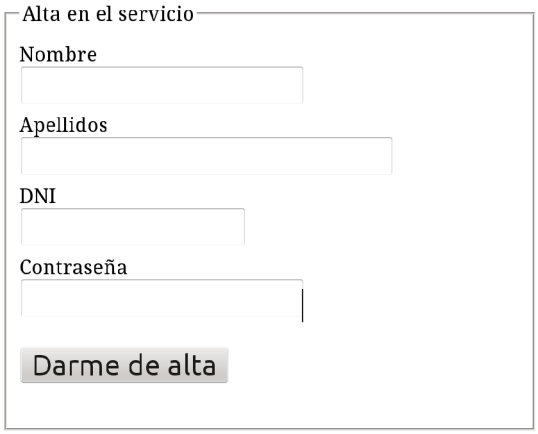
11.1.3 Labels alineadas y formateadas

Los elementos <input> y <label> de los formularios son elementos en línea, por lo que el aspecto que muestran los formularios por defecto, es similar al de la siguiente imagen:



Que será producto del siguiente código: **Ej11.01c-InputsAlineados.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>Inputs Alienadas</title>
<style type="text/css">
<!-- Aquí van las 2 reglas del final de la página -->
</style>
</head>
<body>
<form action="maneja_formulario.php" method="post">
<fieldset>
  <legend>Alta en el servicio</legend>
  <label for="nombre">Nombre</label>
  <input type="text" id="nombre" />
  <label for="apellidos">Apellidos</label>
  <input type="text" id="apellidos" size="50" />
  <label for="dni">DNI</label>
  <input type="text" id="dni" size="10" maxlength="9" />
  <label for="clave">Contraseña</label>
  <input type="password" id="clave" />
  <input class="btn" type="submit" value="Darme de alta" />
</fieldset>
</form>
</body>
</html>
```



Aprovechando los elementos <label>, se pueden aplicar unos estilos CSS sencillos que permitan mostrar el formulario con el aspecto de la siguiente imagen:

En primer lugar, se muestran los elementos <label> como elementos de bloque, para que añadan una separación para cada campo del formulario. Además, se añade un margen superior para no mostrar juntas todas las filas del formulario:

```
label {       display: block;
              margin: .5em 0 0 0;
}
```

El botón del formulario también se muestra como un elemento de bloque y se le añade un margen para darle el aspecto final deseado:

```
.btn {        display: block;
              margin: 1em 0;
}
```

Alta en el servicio

Nombre

Apellidos

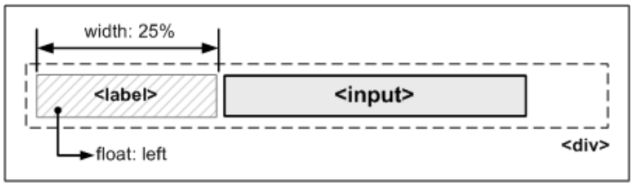
DNI

Contraseña

Darme de alta

En ocasiones, es más útil mostrar todos los campos del formulario con su <label> alineada a la izquierda y el campo del formulario a la derecha de cada <label>, como muestra la siguiente imagen:

Para mostrar un formulario tal y como aparece en la imagen anterior no es necesario crear una tabla y controlar la anchura de sus columnas para conseguir una alineación perfecta. Sin embargo, sí que es necesario añadir un nuevo elemento (por ejemplo un) que encierre a cada uno de los campos del formulario (<label> y <input>). El esquema de la solución propuesta es el siguiente:



La modificación del ejemplo anterior quedará del siguiente modo: **Ej11.01d-LabelsAlineados.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>Etiquetas Alienadas</title>
<style type="text/css">
  div {  margin: .4em 0;      }
  div label {  width: 25%;
               float: left;   }
</style>
</head>
<body>
<form action="maneja_formulario.php" method="post">
<fieldset>
  <legend>Alta en el servicio</legend>
  <div>
    <label for="nombre">Nombre</label>
    <input type="text" id="nombre" />
  </div>
  <div>
    <label for="apellidos">Apellidos</label>
    <input type="text" id="apellidos" size="35" />
  </div>
  <div>
    <label for="dni">DNI</label>
    <input type="text" id="dni" size="10" maxlength="9" />
  </div>
  <div>
    <label for="clave">Contraseña</label>
    <input type="password" id="clave" />
  </div>
  <input class="btn" type="submit" value="Darme de alta" />
</fieldset>
</form>
</body>
</html>
```

11.2 Estilos avanzados

11.2.1 Formulario en varias columnas

Los formularios complejos con decenas de campos pueden ocupar mucho espacio en la ventana del navegador. Además del uso de pestañas para agrupar los campos relacionados en un formulario, también es posible mostrar el formulario a dos columnas, para aprovechar mejor el espacio.

Alta en el servicio

Nombre

Apellidos

DNI

Contraseña

Datos de Contacto

Teléfono

Email

Dirección

Código Postal

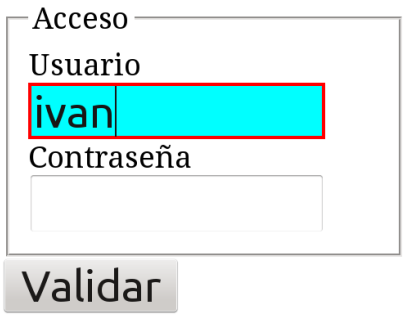
Darme de alta

El código: **Ej11.02a-Form2Columnas.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>Form2Columnas</title>
<style type="text/css">
  form fieldset {
    float: left;
    width: 40%;
  }
</style>
</head>

<body>
<form action="maneja_formulario.php" method="post">
<fieldset>
  <legend>Alta en el servicio</legend>
    <label for="nombre">Nombre</label> <br/>
    <input type="text" id="nombre" /><br/>
    <label for="apellidos">Apellidos</label><br/>
    <input type="text" id="apellidos" size="20" /><br/>
    <label for="dni">DNI</label><br/>
    <input type="text" id="dni" size="10" maxlength="9" /><br/>
    <label for="clave">Contraseña</label><br/>
    <input type="password" id="clave" />
</fieldset>
<fieldset>
  <legend>Datos de Contacto</legend>
    <label for="tfno">Teléfono</label><br/>
    <input type="text" id="tfno" /><br/>
    <label for="email">Email</label><br/>
    <input type="text" id="email" size="20" /><br/>
    <label for="dir">Dirección</label><br/>
    <input type="text" id="dni" size="20" /><br/>
    <label for="cp">Código Postal</label><br/>
    <input type="text" id="cp" size="5" maxlength="9" /><br/>
</fieldset>
    <input class="btn" type="submit" value="Darme de alta" />
</form>
</body>
</html>
```

11.2.2 Resaltar el campo seleccionado

Una de las mejoras más útiles para los formularios HTML consiste en resaltar de alguna forma especial el campo en el que el usuario está introduciendo datos. Para ello, CSS define la pseudo-clase :focus, que permite aplicar estilos especiales al elemento que en ese momento tiene el foco o atención del usuario.



La siguiente imagen muestra un formulario que resalta claramente el campo en el que el usuario está introduciendo la información:

El código: **Ej11.02b-InputFocus.html**

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>Input Focus</title>
<style type="text/css">
  input:focus {   border: 2px solid red;
                  background: cyan;   }
</style>
</head>

<body>
<form action="maneja_formulario.php" method="post">
<fieldset>
  <legend>Accesp</legend>
    <label for="usuario">Usuario</label> <br/>
    <input type="text" id="usuario" /><br/>
    <label for="clave">Contraseña</label><br/>
    <input type="password" id="clave" />
</fieldset>
    <input type="submit" value="Validar" />
</form>
</body>
</html>
```

11.2.3 Mensaje de Ayuda

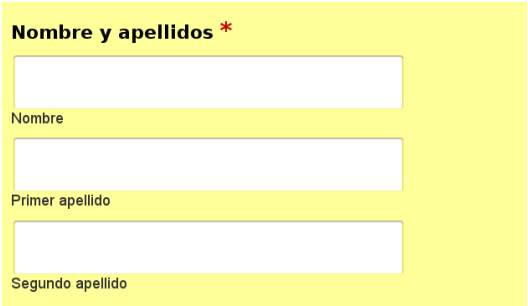
Una mejora al formulario anterior puede ser la inclusión de un mensaje de ayuda que únicamente aparezca al tener el ratón por encima del formulario. Para ello usaremos pseudo-clase :hover aplicado al input. El código será a partir del anterior: **Ej11.02c-MensajeAyuda.html**

```
<!DOCTYPE html>
...
<title>Mensaje Ayuda</title>
<style type="text/css">
   input:focus {  border: 2px solid red;
                  background: cyan;   }
   span {      display: none;                }
   input:focus + span,  input:hover + span
         {     display: inline;
               position: relative;
               left: 10px;                   }
</style>
</head>
<body>
<form action="maneja_formulario.php" method="post">
   …
      <label for="clave">Contraseña</label><br/>
      <input type="password" id="clave" />
      <span> La contraseña debe tener al menos 8 caracteres</span>
   ...
</body>
</html>
```

Ejercicio 12

1) Aplicar las reglas CSS necesarias para crear un formulario con el siguiente aspecto:

2) Cuando el usuario pasa el ratón por encima de cada grupo de elementos de formulario (es decir, por encima de cada <li>) se debe modificar su color de fondo (sugerencia: color amarillo claro #FF9). Además, cuando el usuario se posiciona en un cuadro de texto, se debe modificar su borde para resaltar el campo que está activo cada momento (sugerencia: color amarillo #E6B700):



3) Utilizando el menor número de reglas CSS, cambiar el aspecto del formulario para que se muestre en una sola columna.
4) Cuando el usuario pasa el ratón por encima de un grupo de elementos de formulario (es decir, por encima de cada <li>) se debe mostrar el mensaje de ayuda asociado. Añadir las reglas CSS necesarias para que el formulario tenga el aspecto de la siguiente imagen:

Mensajes (por Secciones):

- No te olvides de escribir también tu segundo apellido (Nombre a Apellidos)
- Asegúrate de que sea válido (Email)
- El código postal es imprescindible para poder recibir los pedidos (Dirección)
- Sin prefijo de país y sin espacios en blanco (Teléfono)

12 LAYOUT O ESTRUCTURA

El diseño de las páginas web habituales se divide en bloques: cabecera, menú, contenidos y pie de página. Visualmente, los bloques se disponen en varias filas y columnas. Por este motivo, hace varios años la estructura de las páginas HTML se definía mediante tablas.

El desarrollo de CSS ha permitido que se puedan realizar los mismos diseños sin utilizar tablas HTML. Las principales ventajas de diseñar la estructura de las páginas web con CSS en vez de con tablas HTML son las siguientes:

- **Mantenimiento**: una página diseñada exclusivamente con CSS es mucho más fácil de mantener que una página diseñada con tablas. Cambiar el aspecto de una página creada con CSS es tan fácil como modificar unas pocas reglas en las hojas de estilos. Sin embargo, realizar la misma modificación en una página creada con tablas supone un esfuerzo muy superior y es más probable cometer errores.
- **Accesibilidad**: las páginas creadas con CSS son más accesibles que las páginas diseñadas con tablas. De hecho, los navegadores que utilizan las personas discapacitadas (en especial las personas invidentes) pueden tener dificultades con la estructura de las páginas complejas creadas con tablas HTML. No obstante, diseñar una página web exclusivamente con CSS no garantiza que la página sea accesible.

- **Velocidad de carga:** el código HTML de una página diseñada con tablas es mucho mayor que el código de la misma página diseñada exclusivamente con CSS, por lo que tarda más tiempo en descargarse. En cualquier caso, si el usuario accede al sitio con una conexión de banda ancha y la página es de un tamaño medio o reducido, las diferencias son casi imperceptibles.
- **Semántica**: aunque resulta obvio, las tablas HTML sólo se deben utilizar para mostrar datos cuya información sólo se entiende en forma de filas y columnas. Utilizar tablas para crear la estructura completa de una página es tan absurdo como utilizar por ejemplo la etiqueta <ul> para crear párrafos de texto.

Por estos motivos, la estructura basada en tablas ha dado paso a la estructura basada exclusivamente en CSS. Aunque crear la estructura de las páginas sólo con CSS presenta en ocasiones retos importantes, en general es más sencilla y flexible. En este capítulo se muestra cómo crear algunas de las estructuras o layouts más habituales de los diseños web utilizando exclusivamente CSS.

12.1 Centrar una página horizontalmente

A medida que aumenta el tamaño y la resolución de las pantallas de ordenador, se hace más difícil diseñar páginas que se adapten al tamaño de la ventana del navegador. El principal reto que se presenta con resoluciones superiores a 1024 x 768 píxel, es que las líneas de texto son demasiado largas como para leerlas con comodidad. Por ese motivo, normalmente se opta por diseños con una anchura fija limitada a un valor aceptable para mantener la legibilidad del texto.

Por otra parte, los navegadores alinean por defecto las páginas web a la izquierda de la ventana. Cuando la resolución de la pantalla es muy grande, la mayoría de páginas de anchura fija alineadas a la izquierda parecen muy estrechas y provocan una sensación de vacío.

La solución más sencilla para evitar los grandes espacios en blanco consiste en crear páginas con una anchura fija adecuada y centrar la página horizontalmente respecto de la ventana del navegador. Utilizando la propiedad margin de CSS, es muy sencillo centrar una página web horizontalmente. La solución consiste en agrupar todos los contenidos de la página en un elemento y asignarle a ese unos márgenes laterales automáticos.

El que encierra los contenidos se suele llamar contenedor (en inglés wrapper o container):

```
#contenedor {
  width: 300px;
  margin: 0 auto;
}

<body>
  <div id="contenedor">
    <h1>Lorem ipsum dolor sit amet</h1>
    ...
  </div>
</body>
```

Como se sabe, el valor 0 auto significa que los márgenes superior e inferior son iguales a 0 y los márgenes laterales toman un valor de auto. Cuando se asignan márgenes laterales automáticos a un elemento, los navegadores centran ese elemento respecto de su elemento padre. En este ejemplo, el elemento padre del es la propia página (el elemento <body>), por lo que se consigue centrar el elemento respecto de la ventana del navegador.

Modificando ligeramente el código CSS anterior se puede conseguir un diseño dinámico o líquido (también llamado fluido) que se adapta a la anchura de la ventana del navegador y permanece siempre centrado:

```
#contenedor {
  width: 70%;
  margin: 0 auto;
}
```

Estableciendo la anchura del elemento contenedor mediante un porcentaje, su anchura se adapta de forma continua a la anchura de la ventana del navegador. De esta forma, si se reduce la anchura de la ventana del navegador, la página se verá más estrecha y si se maximiza la ventana del navegador, la página se verá más ancha.

12.2 Centrar una página verticalmente

Cuando se centra una página web de forma horizontal, sus márgenes laterales se adaptan dinámicamente de forma que la página siempre aparece en el centro de la ventana del navegador, independientemente de la anchura de la ventana. De la misma forma, cuando se centra una página web verticalmente, el objetivo es que sus contenidos aparezcan en el centro de la ventana del navegador y por tanto, que sus márgenes verticales se adapten de forma dinámica en función del tamaño de la ventana del navegador.

Aunque centrar una página web horizontalmente es muy sencillo, centrarla verticalmente es mucho más complicado. Afortunadamente, no es muy común que una página web aparezca centrada de forma vertical. El motivo es que la mayoría de páginas web son más altas que la ventana del navegador, por lo que no es posible centrarlas verticalmente.

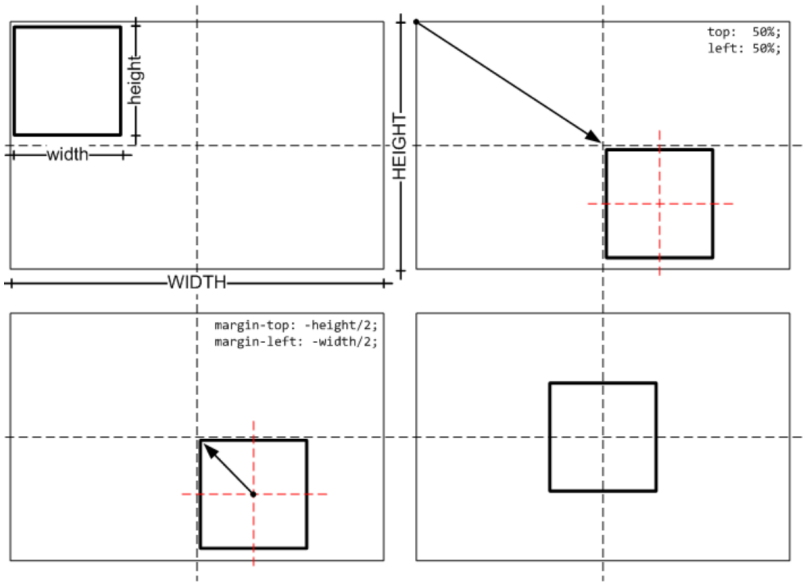
A continuación se muestra la forma de centrar una página web respecto de la ventana del navegador, es decir, centrarla tanto horizontalmente como verticalmente.

Siguiendo el mismo razonamiento que el planteado para centrar la página horizontalmente, se podrían utilizar las siguientes reglas CSS para centrar la página respecto de la ventana del navegador:

```
#contenedor {
  width: 300px;
  height: 250px;
  margin: auto;
}
```

```
 <body>
  <div id="contenedor">
    <h1>Lorem ipsum dolor sit amet</h1>
    ...
  </div>
</body>
```

Si el valor auto se puede utilizar para que los márgenes laterales se adapten dinámicamente, también debería ser posible utilizar el valor auto para los márgenes verticales. Desafortunadamente, la propiedad margin: auto no funciona tal y como se espera para los márgenes verticales y la página no se muestra centrada.



La solución correcta para centrar verticalmente una página web se basa en el posicionamiento absoluto e implica realizar un cálculo matemático sencillo. A continuación se muestra el esquema gráfico de los cuatro pasos necesarios para centrar una página web en la ventana del navegador:

En primer lugar, se asigna una altura y una anchura al elemento que encierra todos los contenidos de la página. En la primera figura, los contenidos de la página tienen una anchura llamada width y una altura llamada height que son menores que la anchura y altura de la ventana del navegador. En el siguiente ejemplo, se supone que tanto la anchura como la altura de la página es igual a 500px:

```
#contenedor {
  width: 500px;
  height: 500px;
}

<body>
  <div id="contenedor">
    <h1>Lorem ipsum dolor sit amet</h1>
    ...
  </div>
</body>
```

Si la página se debe mostrar en el centro de la ventana del navegador, es necesario desplazar hacia arriba y hacia la izquierda los contenidos de la página web. Para determinar el desplazamiento necesario, se realiza un cálculo matemático sencillo. Como se ve en la tercera figura del esquema anterior, el punto central de la página debe desplazarse hasta el centro de la ventana del navegador.

Como se desprende de la imagen anterior, la página web debe moverse hacia arriba una cantidad igual a la mitad de su altura y debe desplazarse hacia la izquierda una cantidad equivalente a la mitad de su anchura. Utilizando las propiedades margin-top y margin-left con valores negativos, la página se desplaza hasta el centro de la ventana del navegador.

```
#contenedor {
  width: 500px;
  height: 500px;

  position: absolute;
  top: 50%;
  left: 50%;

  margin-top: -250px;   /* height/2 = 500px / 2 */
  margin-left: -250px;  /* width/2 = 500px / 2 */
}
```

Con las reglas CSS anteriores, la página web siempre aparece centrada verticalmente y horizontalmente respecto de la ventana del navegador. El motivo es que la anchura/altura de la página son fijas (propiedades width y height), el desplazamiento necesario para centrarla también es fijo (propiedades margin-top y margin-left) y el desplazamiento hasta el centro de la ventana del navegador se calcula dinámicamente gracias al uso de porcentajes en las propiedades top y left.

Para centrar una página sólo verticalmente, se debe prescindir tanto del posicionamiento horizontal como del desplazamiento horizontal:

```
#contenedor {
  width: 500px;
  height: 500px;

  position: absolute;
  top: 50%;

  margin-top: -250px;   /* height/2 = 500px / 2 */
}
```

12.2.1 Centrar una página verticalmente (Opción2)

Una buena razón por lo que muchas personas defienden el uso de tablas para la maquetación de contenidos de un sitio web, es porque éstas permiten el correcto centrado horizontal y vertical de los elementos contenidos dentro de sus celdas. La propiedad vertical-align permite un centrado total junto con un par de valores poco utilizados de display: table y table-cell.

Antes que nada, necesitamos que nuestra superficie esté definida, en este caso al 100% de la pantalla (sin scroll). Esto lo logramos con height: 100% en 2 partes: el html y el contenedor de lo que queremos mostrar:

```
html, body {
     margin: 0;
     padding: 0;
     height: 100%;
}

#contenedor {
     display: table;
     height: 100%;
     width: 100%;
     margin: 0;
   }
```

Nuestro poco utilizado display: table; lo que hace es dejar nuestro #contenedor compacto. Dentro de #contenido puedes encontrar display: table-cell; que se comporta tal como una celda dentro de una tabla (en este caso, #contenido sería una celda dentro de la tabla #contenedor).

Y es por este comportamiento de tablas que vertical-align entra en acción:

```
#content {
      display: table-cell;
      vertical-align: middle;
      position: relative;
}
```

Veamos el ejemplo completo: **Ej12.02a-CentradoTotal.html**

```
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>Centrado Total</title>
<style type="text/css">

html, body {
    margin: 0;
    padding: 0;
    height: 100%;
    }

p {
    font-size: 14px;
   line-height: 150%;
    }

h1 {
    color: #C00;
   text-shadow: #000 1px 1px 2px;
    }

#contenedor {
    display: table;
    height: 100%;
    width: 100%;
    margin: 0;
    }

#contenido {
    display: table-cell;
    vertical-align: middle;
    position: relative;
    }

#caja {
    border: 1px dashed #F00;
    width: 60%;
    margin: 0 auto;
    padding: 0 20px;
    }

</style>
</head>

<body>
  <div id="contenedor">
    <div id="contenido">
      <div id="caja">
        <h1>Centrado Vertical</h1>
        <p>Utilizando la propiedad display: table para centrar el contenido verticalmente.
        Debemos introducir este texto dentro de una caja (que es donde marcaremos el tamaño
        del ancho deseado)
        </p>
      </div><!-- fin caja -->
    </div><!-- fin contenido" -->
  </div>
</body>
</html>
```

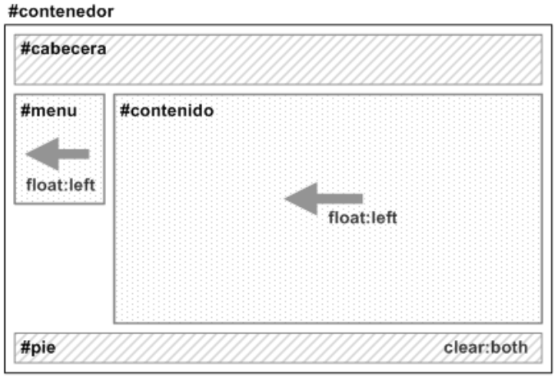
12.3 Estructura o layout

12.3.1 Diseño a 2 columnas con cabecera y pie de página

El objetivo de este diseño es definir una estructura de página con cabecera y pie, un menú lateral de navegación y una zona de contenidos.

La anchura de la página se fija en 700px, la anchura del menú es de 150px y la anchura de los contenidos es de 550px:

La solución CSS se basa en el uso de la propiedad float para los elementos posicionados como el menú y los contenidos y el uso de la propiedad clear en el pie de página para evitar los solapamientos ocasionados por los elementos posicionados con float.



El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

Ej12.03a-Estructura2Col.html

```
#contenedor {  width: 700px;     }
#cabecera {  }
#menu {  float: left; width: 150px; }
#contenido { float: left;width: 550px; }
#pie {  clear: both;       }
<body>
<div id="contenedor">
  <div id="cabecera">
  </div>
  <div id="menu">
  </div>
   <div id="contenido">
  </div>
   <div id="pie">
  </div>
</div>
</body>
```

En los estilos CSS anteriores se ha optado por desplazar tanto el menú como los contenidos hacia la izquierda de la página (float: left). Sin embargo, en este caso también se podría desplazar el menú hacia la izquierda (float:left) y los contenidos hacia la derecha (float: right).

El diseño anterior es de anchura fija, lo que significa que no se adapta a la anchura de la ventana del navegador. Para conseguir una página de anchura variable y que se adapte de forma dinámica a la ventana del navegador, se deben aplicar las siguientes reglas CSS:

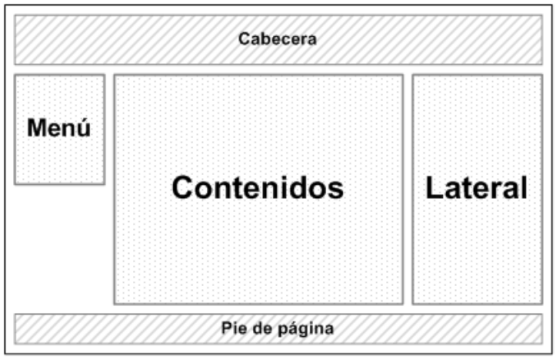
```
#contenedor {}
#cabecera {}
#menu {  float: left;  width: 15%;}
#contenido {  float: left;  width: 85%;}
#pie {  clear: both;}
```

Si se indican la anchuras de los bloques que forman la página en porcentajes, el diseño final es dinámico. Para crear diseños de anchura fija, basta con establecer las anchuras de los bloques en píxel.

Aquí os dejo la estructura completa:

```
<!DOCTYPE html>
<html lang="es">    <!-- Ej12.03a-Estructura2Columnas.html -->
<head>
   <meta charset="UTF-8" />
   <title>2Columnas</title>
   <style type="text/css">
      #contenedor {  width: 700px;

                     /* Hay que quitarlo si la altura es dinámica*/
                     height: 300px;
                     background-color: lavender;
                     margin: 0 auto;
                     border: 1px solid blue;}

      #cabecera{     background-color: HoneyDew;}

      #menu   {      width: 150px;
                     float: left;
                     background-color: cyan;    }

      #contenido {   width: 500px;
                     margin: 10px 25px;
                     float: left;
                     background-color: Beige;}

      #pie    {      padding-top: 10px;
                     clear: both;
                     background-color: HoneyDew;}
   </style>
</head>
<body>
   <div id="contenedor">

      <div id="cabecera">
         <h1> Mi página web</h1>
      </div>

      <div id="menu">
         <ul>
            <p> Menu</p>
            <li> Elemento2</li>
            <li> Elemento3</li>
            <li> Elemento4</li>
            <li> Elemento5</li>
         </ul>
      </div>

      <div id="contenido">
         <h2> Titulo del Contenido</h2>
         <p> </p><p> </p>
      </div>

      <div id="pie">
         <h3> Este es el pie de página</h3>
      </div>

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

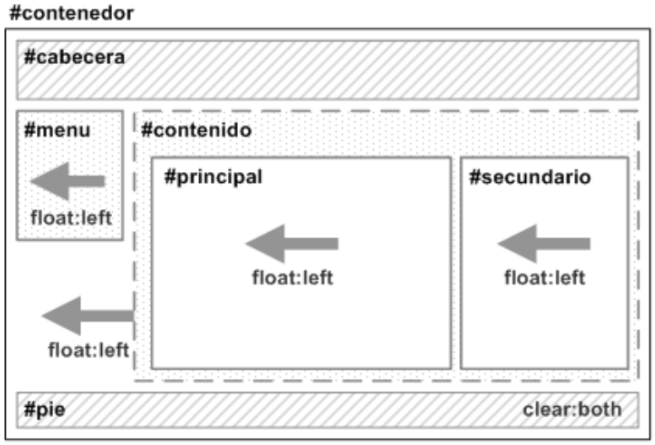
12.3.2 Diseño a 3 columnas con cabecera y pie de página



Además del diseño a dos columnas, el diseño más utilizado es el de tres columnas con cabecera y pie de página.

En este caso, los contenidos se dividen en dos zonas diferenciadas: zona principal de contenidos y zona lateral de contenidos auxiliares (imagen izquierda):

La solución CSS emplea la misma estrategia del diseño a dos columnas y se basa en utilizar las propiedades float y clear (imagen inferior):



El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

Ej12.03b-Estructura3Col.html

```
#contenedor {}
#cabecera {}
#menu {  float: left;width: 15%;}

#contenido {  float: left;
  width: 85%;}
#contenido #principal {
float: left; width: 80%;  }
#contenido #secundario {
float: left; width: 20%;  }
#pie {  clear: both;      }

<body>
<div id="contenedor">
  <div id="cabecera">
  </div>
  <div id="menu">
  </div>

  <div id="contenido">
    <div id="principal">
    </div>
    <div id="secundario">
    </div>
  </div>

  <div id="pie">
  </div>
</div>
</body>
```

El código anterior crea una página con anchura variable que se adapta a la ventana del navegador. Para definir una página con anchura fija, solamente es necesario sustituir las anchuras en porcentajes por anchuras en píxel.

Al igual que sucedía en el diseño a dos columnas, se puede optar por posicionar todos los elementos mediante float: left o se puede utilizar float: left para el menú y la zona principal de contenidos y float: right para el contenedor de los contenidos y la zona secundaria de contenidos.

Aquí os paso una estructura terminada en 3 columnas con algo de contenido:

```
<!DOCTYPE html>
<html lang="es">    <!-- Ej12.03a-Estructura2Columnas.html -->
<head>
   <meta charset="UTF-8" />      <title>3Columnas</title>
   <style type="text/css">
      #contenedor {  width: 90%;
                     /* Hay que quitarlo si la altura es dinámica*/
                     height: 80%;
                     margin: 0 auto;            background-color: lavender;
                     border: 1px solid blue;                              }

      #cabecera {    background-color: HoneyDew;                           }
      #menu {        width: 15%;                float: left;
                     background-color: cyan;                              }

      #contenido {   width: 74%;                margin: 0 5% 5%;
                     float: left;               background-color: green;
                     border: 1px dashed red;                              }

      #contenido #principal {
                     width: 75%;                float: left;
                     background-color: Linen;                             }

      #contenido #secundario {
                     margin-left: 10%;          width: 15%;
                     float: left;               background-color: Salmon; }

      #pie    {      padding-top: 5%;    clear: both;
                     background-color: HoneyDew;                          }
   </style>
</head>
<body>
   <div id="contenedor">
      <div id="cabecera">
         <h1> Mi página web</h1>
      </div>
      <div id="menu">
         <ul>
            <p> Menu</p>
            <li> Elem1</li>              <li> Elem2</li>
            <li> Elem3</li>              <li> Elem4</li>
         </ul>
      </div>
      <div id="contenido">
         <div id="principal">
            <h2> Título del Contenido</h2>
            <p> </p><p> </p>
         </div>
         <div id="secundario">
            <h3>Enlaces</h3>
            <p><a href="#">Google</a></p>
            <p><a href="#">Yahoo</a></p>
         </div>
      </div>
      <div id="pie">
         <h3> Este es el pie de página</h3>
      </div>
   </div>
</body>
</html>
```

Ejercicio 13

Determinar las reglas CSS necesarias para mostrar la página HTML como se muestra en la siguiente imagen:

A continuación se indica una propuesta de los pasos que se pueden seguir para obtener el aspecto final deseado:

- Añadir los estilos básicos de la página (tipo de letra Verdana, color de letra #192666, imagen de fondo llamada fondo.gif, color de fondo #F2F5FE).
- Definir la estructura básica de la página: anchura fija de 760 píxel, centrada en la ventana del navegador, cabecera y pie, columna central de contenidos de anchura 530 píxel y columna secundaria de contenidos de 200 píxel de anchura.
- La cabecera tiene una altura de 100 píxel y una imagen de fondo llamada cabecera.jpg.

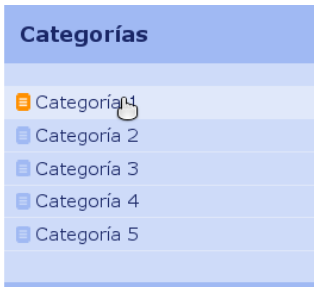
- Los elementos del menú de navegación tienen un color de fondo #253575, un color de letra #B5C4E3. Cuando el ratón pasa por encima de cada elemento, su color de fondo cambia a #31479B. Los elementos seleccionados se muestran con un color de fondo blanco y un color de letra #FF9000:

Con la ayuda de las imágenes que se proporcionan, mostrar cada uno de los artículos de contenido con el estilo que se muestra en la siguiente imagen:

Añadir los estilos adecuados para mostrar los elementos de la columna secundaria de contenidos con el siguiente aspecto.

- Sobre mi
- Categorías
- Archivo

En la zona inferior incluiremos las categorías y el archivo.
Obsérvese como cambia cada elemento al pasar el ratón por encima de ellos.



Las imágenes a emplear se encuentran en:
http://www.librosweb.es/ejercicios/css/ejercicio13/imagenes.zip

Como firma incluiremos nuestro nombre y el año de realización de la práctica en el pie de página.:

© 2013 Iván Rodríguez

12.4 Alturas/anchuras máximas y mínimas

Cuando se diseña la estructura de una página web, se debe tomar la decisión de optar por un diseño de anchura fija o un diseño cuya anchura se adapta a la anchura de la ventana del navegador.

Sin embargo, la mayoría de las veces sería conveniente una solución intermedia: que la anchura de la página sea variable y se adapte a la anchura de la ventana del navegador, pero respetando ciertos límites. En otras palabras, que la anchura de la página no sea tan pequeña como para que no se puedan mostrar correctamente los contenidos y tampoco sea tan ancha como para que las líneas de texto no puedan leerse cómodamente.

CSS define cuatro propiedades que permiten limitar la anchura y altura mínima y máxima de cualquier elemento de la página. Las propiedades son max-width, min-width, max-height y min-height.

Propiedad	**max-width**
Valores	<medida> | <porcentaje> | none | inherit
Se aplica a	Todos los elementos salvo filas y grupos de filas de tablas
Valor inicial	none
Descripción	Permite definir la anchura máxima de un elemento

Propiedad	**min-width**
Valores	<medida> | <porcentaje> | inherit
Se aplica a	Todos los elementos salvo filas y grupos de filas de tablas
Valor inicial	0
Descripción	Permite definir la anchura mínima de un elemento

Propiedad	**max-height**
Valores	<medida> | <porcentaje> | none | inherit
Se aplica a	Todos los elementos salvo columnas y grupos de columnas de tablas
Valor inicial	none
Descripción	Permite definir la altura máxima de un elemento

Propiedad	**min-height**
Valores	<medida> | <porcentaje> | inherit
Se aplica a	Todos los elementos salvo columnas y grupos de columnas de tablas
Valor inicial	0
Descripción	Permite definir la altura mínima de un elemento

De esta forma, para conseguir un diseño de anchura variable pero controlada, se podrían utilizar reglas CSS como la siguiente:

```
#contenedor {
  min-width: 500px;
  max-width: 900px;
}
```

Las propiedades que definen la altura y anchura máxima y mínima se pueden aplicar a cualquier elemento, aunque solamente suelen utilizarse para estructurar la página. En general, las propiedades más utilizadas son max-width y min-width, ya que no es muy habitual definir alturas máximas y mínimas.

12.5 Estilos avanzados

En general, la columna de los contenidos es la más larga y la columna de navegación es la más corta. El principal inconveniente de los diseños mostrados anteriormente es que no se puede garantizar que todas las columnas se muestren con la misma altura.

Si las columnas tienen algún color o imagen de fondo, este comportamiento no es admisible, ya que se vería que alguna columna no llega hasta el final de la columna más larga y el diseño final parecería inacabado.

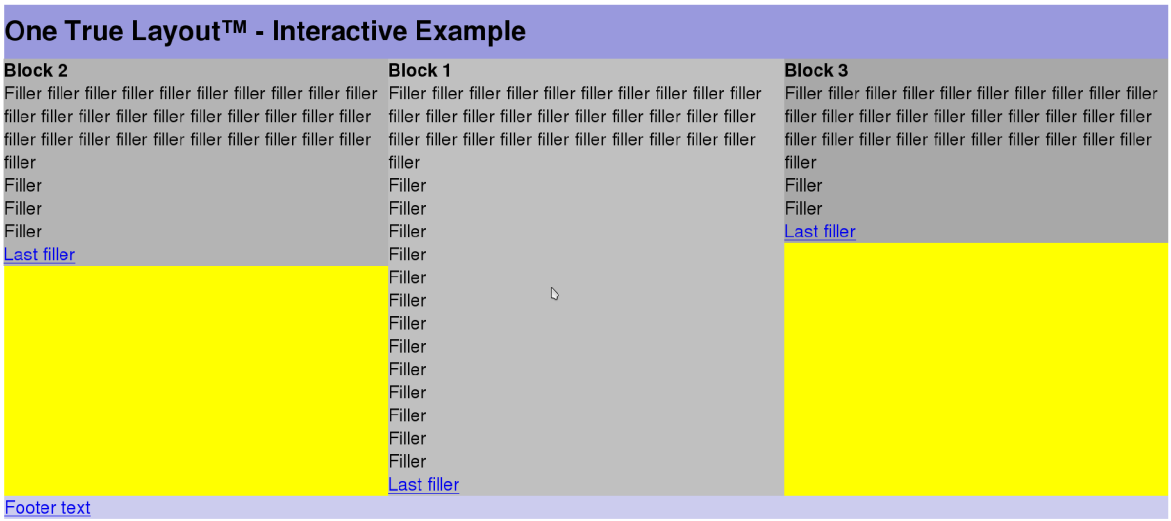
Desde la aparición de este problema se han presentado numerosas soluciones. La más conocida es la técnica faux columns ("columnas falsas") y que simula el color/imagen de fondo de las columnas laterales mediante la imagen de fondo de la columna central de contenidos.

La técnica fue presentada originalmente por Dan Cederholm en su célebre artículo "Faux Columns" (http://alistapart.com/articles/fauxcolumns/).

Más recientemente se ha presentado el proyecto "In Search of the One True Layout" que busca definir una serie de técnicas que permitan crear de forma sencilla cualquier estructura de página basada en columnas.

La página principal del proyecto se puede encontrar en:
http://www.positioniseverything.net/articles/onetruelayout/

Además, está disponible una herramienta interactiva para crear diseños basados en columnas con la posibilidad de definir el número de columnas, su anchura y obligar a que todas las columnas muestren la misma altura:


One True Layout™ - Interactive Example
Block 2
Filler filler filler

13 OTROS

13.1 Propiedades shorthand

Las propiedades de tipo "shorthand" son propiedades de CSS que permiten establecer de forma simultánea el valor de varias propiedades diferentes pero relacionadas. El uso de las propiedades "shorthand" es muy extendido, ya que permiten crear hojas de estilos más compactas.

A continuación se incluye a modo de referencia todas las propiedades de tipo "shorthand" que se han mostrado anteriormente.

Propiedad	**font**
Valores	((<font-style> || <font-variant> || <font-weight>)? <font-size> (/ <line-height>)? <font-family>) | caption | icon | menu | message-box | small-caption | status-bar | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Permite indicar de forma directa todas las propiedades de la tipografía de un texto

Propiedad	**margin**
Valores	(<medida> | <porcentaje> | auto) {1, 4} | inherit
Se aplica a	Todos los elementos salvo algunos casos especiales de elementos mostrados como tablas
Valor inicial	-
Descripción	Establece de forma directa todos los márgenes de un elemento

Propiedad	**padding**
Valores	(<medida> | <porcentaje>){1, 4} | inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	-
Descripción	Establece de forma directa todos los rellenos de los elementos

Propiedad	**border**
Valores	(<medida_borde> || <color_borde> || <estilo_borde>) | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de todos los bordes de los elementos

Propiedad	**background**
Valores	(<background-color> || <background-image> || <background-repeat> || <background-attachment> || <background-position>) | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece todas las propiedades del fondo de un elemento

Propiedad	**list-style**
Valores	(<list-style-type> || <list-style-position> || <list-style-image>) | inherit
Se aplica a	Elementos de una lista
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultanea todas las opciones de una lista

13.2 Versión para imprimir

La mayoría de sitios web de calidad ofrecen al usuario la posibilidad de imprimir sus contenidos mediante una versión específica para impresora de cada página.

Estas versiones optimizadas para impresora eliminan los contenidos superfluos, modifican o eliminan las imágenes y colores de fondo y sobre todo, optimizan los contenidos de texto para facilitar su lectura.

CSS simplifica al máximo la creación de una versión para imprimir gracias al concepto de los medios CSS. Como se sabe, los estilos CSS que se aplican a los contenidos pueden variar en función del medio a través del que se acceden (pantalla, televisor, móvil, impresora, etc.)

De esta forma, realizar una versión para imprimir de una página HTML es tan sencillo como crear unas cuantas reglas CSS que optimicen sus contenidos para conseguir la mejor impresión.

El sitio web A List Apart es un excelente ejemplo de cómo los sitios web de calidad crean versiones específicas para impresora de las páginas web originales.

Crear una versión para imprimir similar a la mostrada en la web anterior es una tarea que no lleva más de unos pocos minutos. Las reglas CSS de la versión para imprimir se aplican solamente al medio print. Por lo tanto, en primer lugar se crea una nueva hoja de estilos y al enlazarla se especifica que sólo debe aplicarse en las impresoras:

```
<link rel="stylesheet" type="text/css" href="/css/imprimir.css" media="print" />
```

Normalmente, las hojas de estilos para la pantalla se aplican a todos los medios (por utilizar el valor media="all" al enlazarla o por no indicar ningún valor en el atributo media). Por este motivo, cuando se imprime una página se aplican los mismos estilos que se aplican al visualizarla en la pantalla.

Aprovechando este comportamiento, las hojas de estilos para impresoras son muy sencillas, ya que sólo deben modificar algunos estilos aplicados en el resto de hojas de estilos.

Por este motivo, normalmente las hojas de estilos para impresora se construyen siguiendo los pasos que se muestran a continuación:

1) Ocultar los elementos que no se van a imprimir:

```
#cabecera, #menu, #lateral, #comentarios {
  display: none !important;
}
```

Los bloques (normalmente encerrados en elementos de tipo) que no se van a imprimir se ocultan con la propiedad display y su valor none. La palabra clave !important aumenta la prioridad de esta regla CSS y más adelante se explica su significado.

2) Corregir la estructura de la página:

```
body, #contenido, #principal, #pie {
  float: none !important;
  width: auto !important;
  margin:  0 !important;
  padding: 0 !important;
}
```

Normalmente, las páginas web complejas están formadas por varias columnas posicionadas mediante la propiedad float. Si al imprimir la página se eliminan las columnas laterales, es conveniente reajustar la anchura y el posicionamiento de la zona de contenidos y de otras zonas que sí se van a imprimir.

3) Modificar los colores y tipos de letra:

```
body { color: #000; font: 100%/150% Georgia, "Times New Roman", Times, serif; }
```

Aunque el uso de impresoras en color es mayoritario, suele ser conveniente imprimir todo el texto de las páginas de color negro, para ahorrar costes y para aumentar el contraste cuando se imprime sobre hojas de color blanco. También suele ser conveniente modificar el tipo de letra y escoger uno que facilite al máximo la lectura del texto.

13.2.1 Imprimiendo los enlaces

Uno de los principales problemas de las páginas HTML impresas es la pérdida de toda la información relativa a los enlaces. En principio, imprimir los enlaces de una página es absurdo porque no se pueden utilizar en el medio impreso.

Sin embargo, lo que puede ser realmente útil es mostrar al lado de un enlace la dirección a la que apunta. De esta forma, al imprimir la página no se pierde la información relativa a los enlaces.

CSS incluye una propiedad llamada content que permite crear nuevos contenidos de texto para añadirlos a la página HTML. Si se combina la propiedad content y el pseudo-elemento :after, es posible insertar de forma dinámica la dirección a la que apunta un enlace justo después de su texto:

```
a:after {
  content: " (" attr(href) ") ";
}
```

El código CSS anterior añade después de cada enlace de la página un texto formado por la dirección a la que apunta el enlace mostrada entre paréntesis. Si se quiere añadir las direcciones antes de cada enlace, se puede utilizar el pseudo-elemento :before:

```
a:before {
  content: " (" attr(href) ") ";
}
```

Veamos un ejemplo: **Ej13.02a-MostrarEnlaces.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Mostrar Enlaces</title>
  <link rel="stylesheet" type="text/css" href="Ej13.01a-ImprimirEnlaces.css"
media="print" />
  <style></style>
</head>
<body>
  <a href="http://www.google.es">Google</a>
</body>
</html>
```

Ej13.01a-MostrarEnlaces.css

```
a:after {
  content: " (" attr(href) ") ";
}
```

13.3 Personalizar el cursor

CSS no permite modificar los elementos propios del navegador o de la interfaz de usuario del sistema operativo. Sin embargo, el puntero del ratón es una excepción muy importante, ya que se puede modificar mediante la propiedad cursor.

Propiedad	**cursor**
Valores	((<url> ,)* (auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize | text | wait | help | progress)) | inherit
Se aplica a	Todos los elementos
Valor inicial	auto
Descripción	Permite personalizar el puntero del ratón

La propiedad cursor no sólo permite seleccionar un puntero entre los disponibles en el sistema operativo (flecha, mano, reloj de arena, redimensionar, etc.) sino que incluso permite indicar la URL de una imagen que se quiere mostrar como puntero personalizado.

Se pueden indicar varias URL para que CSS intente cargar varias imágenes: si la primera imagen del puntero no se carga o no la soporta el navegador, se pasa a la siguiente imagen y así sucesivamente hasta que se pueda cargar alguna imagen.

El siguiente ejemplo muestra el caso de un puntero definido con varias URL y un valor estándar:

Ej13.03a-CambiarPuntero.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Imprimir Enlaces</title>
  <style type="text/css">
    a:link, a:visited { cursor: url("../img/Puntero.svg"), url(puntero.cur), wait; }

  </style>
</head>
<body>
  <a href="http://www.google.es">Google</a>
</body>
</html>
```

NOTA: he probado con varios archivos svg y cur y ninguno funciona.

Si el navegador soporta las imágenes en formato SVG, el puntero del ratón cambia su aspecto por la imagen puntero.svg. Si el navegador no soporta el formato SVG, intenta cargar la siguiente URL que define un puntero en formato .cur. Si no se puede cargar correctamente, se mostraría el valor preestablecido pointer.

El puntero personalizado más utilizado es la opción cursor: pointer y cursor: hand que muestra en el puntero una mano que puede pinchar sobre el elemento. Otro puntero muy utilizado es cursor: move que permite indicar en las aplicaciones web dinámicas los elementos de la página que se pueden mover.

Se puede ver un ejemplo de cada uno de los punteros y la compatibilidad con los diferentes navegadores en la siguiente página:
http://www.echoecho.com/csscursors.htm

13.4 Hacks y filtros

Los diferentes navegadores y las diferentes versiones de cada navegador incluyen defectos y carencias en su implementación del estándar CSS 2.1. Algunos navegadores no soportan ciertas propiedades, otros las soportan a medias y otros ignoran el estándar e incorporan su propio comportamiento.

De esta forma, diseñar una página compleja que presente un aspecto homogéneo en varios navegadores y varias versiones diferentes de cada navegador es una tarea que requiere mucho esfuerzo. Para facilitar la creación de hojas de estilos homogéneas, se han introducido los filtros y los hacks.

A pesar de que utilizar filtros y hacks es una solución poco ortodoxa, en ocasiones es la única forma de conseguir que una página web muestre un aspecto idéntico en cualquier navegador.

En primer lugar, los filtros permiten definir u ocultar ciertas reglas CSS para algunos navegadores específicos. Los filtros se definen aprovechando los errores de algunos navegadores (sobre todo los antiguos) a la hora de procesar las hojas de estilos.

Un caso especial de filtro son los comentarios condicionales, que es un mecanismo propietario del navegador Internet Explorer. Los comentarios condicionales permiten incluir hojas de estilos o definir reglas CSS específicamente para una versión de Internet Explorer.

El siguiente ejemplo carga la hoja de estilos basico_ie.css solamente para los navegadores de tipo Internet Explorer:

```
<!--[if IE]>
  <style type="text/css">
    @import ("basico_ie.css");
  </style>
<![endif]-->
```

Los navegadores que no son Internet Explorer ignoran las reglas CSS anteriores ya que interpretan el código anterior como un comentario de HTML (gracias a los caracteres <!-- y -->) mientras que los navegadores de la familia Internet Explorer lo interpretan como una instrucción propia y válida.

El filtro [if IE] indica que esos estilos CSS sólo deben tenerse en cuenta si el navegador es cualquier versión de Internet Explorer. Utilizando comentarios condicionales, también es posible incluir reglas CSS para versiones específicas de Internet Explorer:

```
<!--[if gte IE 6]>
  <style type="text/css">
    @import ("basico_ie6.css");
  </style>
<![endif]-->
```

El anterior ejemplo solamente carga la hoja de estilos basico_ie6.css si el navegador es la versión 6 o superior de Internet Explorer, ya que gte se interpreta como "greater than or equal" ("igual o mayor que"). Otros valores disponibles son gt ("greater than" o "mayor que"), lt ("less than" o "menor que") y lte ("less than or equal" o "igual o menor que").

```
<!--[if gt IE 7]>
  Mayor que Internet Explorer 7
<![endif]-->
```

```
<!--[if gte IE 7]>
  Mayor o igual que Internet Explorer 7
<![endif]-->

<!--[if lt IE 8]>
  Menor que Internet Explorer 8
<![endif]-->

<!--[if lte IE 7]>
  Igual o menor que Internet Explorer 7
<![endif]-->
```

Una de las mejores listas actualizadas con todos los filtros disponibles para los navegadores de los diferentes sistemas operativos se puede encontrar en http://centricle.com/ref/css/filters/

Por otra parte, los hacks permiten forzar el comportamiento de un navegador para que se comporte tal y como se espera. Se trata de una forma poco elegante de crear las hojas de estilos y los hacks se pueden considerar pequeños parches y chapuzas que permiten que la hoja de estilos completa se muestre tal y como se espera.

Uno de los hacks más conocidos y utilizados es el llamado * html. Todas las propiedades CSS que se establezcan mediante el selector * html son interpretadas exclusivamente por el navegador Internet Explorer 6 y sus versiones anteriores:

```
div {
  border-bottom: 1px dotted #000;
}
* html div {
  border-bottom: 1px solid #000;
}
```

El ejemplo anterior utiliza el hack * html para mostrar un borde inferior punteado en los en todos los navegadores salvo Internet Explorer 6. Como en este navegador no se muestran correctamente los bordes punteados de 1 píxel de anchura, se decide mostrar un borde formado por una línea continua.

El otro hack más conocido y utilizado por su sencillez es el "underscore hack". Las propiedades cuyos nombres se indiquen con un guión bajo por delante, sólo son interpretadas por el navegador Internet Explorer 6 y sus versiones anteriores:

```
#menu {
  position: fixed;
  _position: static;
}
```

Los navegadores más modernos soportan el valor fixed para la propiedad position, pero Internet Explorer 6 no la soporta. Por este motivo, la regla CSS anterior establece el valor de la propiedad position y seguidamente define la propiedad _position.

Los navegadores que siguen los estándares rechazan la propiedad _position, ya que su nombre no se corresponde con ninguna propiedad válida de CSS. Internet Explorer 6 y las versiones anteriores, consideran correcta tanto position como _position, por lo que el valor utilizado será el que se haya definido en último lugar (static en este caso).

Una de las mejores listas actualizadas con los hacks más útiles para varios navegadores de diferentes sistemas operativos se puede encontrar en: http://css-discuss.incutio.com/?page=CssHack

13.5 Prioridad de las declaraciones CSS

Además de las hojas de estilos definidas por los diseñadores, los navegadores aplican a cada página otras dos hojas de estilos: la del navegador y la del usuario.

La hoja de estilos del navegador es la primera que se aplica y se utiliza para establecer el estilo inicial por defecto a todos los elementos HTML (tamaños de letra iniciales, decoración del texto, márgenes entre elementos, etc.)

Además de la hoja de estilos del navegador, cada usuario puede crear su propia hoja de estilos y aplicarla automáticamente a todas las páginas que visite con su navegador. Se trata de una opción muy útil para personas discapacitadas visualmente, ya que pueden aumentar el contraste y el tamaño del texto de todas las páginas para facilitar su lectura.

La forma en la que se indica la hoja de estilo del usuario es diferente en cada navegador. A continuación se muestra cómo se hace en los navegadores más populares:

Internet Explorer

1. Pincha sobre el menú Herramientas y después sobre la opción Opciones de Internet
2. En la pestaña General que se muestra, pulsa sobre el botón Accesibilidad que se encuentra dentro de la sección Apariencia
3. En la nueva ventana que aparece, activa la opción Formatear los documentos con mi hoja de estilos y selecciónala pulsando sobre el botón Examinar...
4. Pulsa Aceptar hasta volver al navegador

Firefox

1. Guarda tu hoja de estilos en un archivo llamado userContent.css
2. Entra en el directorio de tu perfil de usuario de Firefox. En los sistemas operativos Windows este directorio se encuentra normalmente en C:\Documents and Settings\[tu_usuario_de_windows]\Datos de programa\Mozilla\Firefox\Profiles\[cadena_aleatoria_de_letras_y_numeros].default
3. Copia la hoja de estilos userContent.css en el directorio chrome de tu perfil
4. Reinicia el navegador para que se apliquen los cambios

Safari

1. Pincha sobre el menú Editar y después sobre la opción Preferencias
2. Selecciona la sección Avanzado
3. Pincha sobre la lista desplegable llamada Hoja de estilos y selecciona la opción Otra...
4. En la ventana que aparece, selecciona tu hoja de estilos

Opera

1. Pincha sobre el menú Herramientas y después sobre la opción Preferencias
2. Selecciona la pestaña Avanzado y pulsa sobre el botón Opciones de estilo...
3. Pulsa sobre el botón Seleccionar... para seleccionar la hoja de estilos
4. Pulsa Aceptar hasta volver al navegador

El orden normal en el que se aplican las hojas de estilo es el siguiente:

Por tanto, las reglas que menos prioridad tienen son las del CSS por defecto de los navegadores, ya que son las primeras que se aplican. A continuación se aplican las reglas definidas por los usuarios y por último se aplican las reglas CSS definidas por el diseñador, que por tanto son las que más prioridad tienen.

Además de estas hojas de estilos, CSS define la palabra reservada !important para controlar la prioridad de las declaraciones de las diferentes hojas de estilos.

Si a una declaración CSS se le añade la palabra reservada !important, se aumenta su prioridad. El siguiente ejemplo muestra el uso de !important:

```
p {
  color: red !important;
  color: blue;
}
```

Si la primera declaración no tuviera añadido el valor !important, el color de los párrafos sería azul, ya que en el caso de declaraciones de la misma importancia, prevalece la indicada en último lugar. Sin embargo, como la primera declaración se ha marcado como de alta prioridad (gracias al valor ! important), el color de los párrafos será el rojo.

El valor !important no sólo afecta a las declaraciones simples, sino que varía la prioridad de las hojas de estilo. Cuando se indican declaraciones de alta prioridad, el orden en el que se aplican las hojas de estilo es el siguiente:

Los estilos del usuario marcados como !important tienen más prioridad que los estilos marcados como !important en la hoja de estilos del diseñador. De esta forma, ninguna página web puede sobreescribir o redefinir ninguna propiedad de alta prioridad establecida por el usuario.

Si se aplica el valor !important a una propiedad de tipo "shorthand", se interpreta como si se hubiera aplicado el valor !important a cada una de las propiedades individuales.

13.6 Validador

La validación del código CSS y de las reglas que lo forman es un concepto similar a la validación de documentos XHTML. La validación es un mecanismo que permite comprobar que el código CSS creado cumple las reglas de la sintaxis del lenguaje CSS y que por tanto es una hoja de estilos válida para aplicarla a cualquier documento XHTML.

La validación suele ser útil cuando se producen errores en los estilos definidos o comportamientos no deseados al aplicar las reglas CSS. En muchas ocasiones, los errores se producen porque el navegador está ignorando algunas reglas que contienen propiedades mal definidas o errores de sintaxis.

El W3C (World Wide Web Consortium) dispone de un validador online que permite validar reglas CSS sueltas, páginas XHTML con CSS incluído y archivos CSS independientes. El validador se puede acceder en http://jigsaw.w3.org/css-validator/.

13.7 Recomendaciones generales sobre CSS

13.7.1 Atributos ID y class

El atributo id se emplea para identificar a cada elemento HTML, por lo que los identificadores deben ser únicos en una misma página. En otras palabras, dos elementos HTML diferentes de una misma página no pueden tener un mismo valor en el atributo id.

Por otra parte, el atributo class se emplea para indicar la clase o clases a las que pertenece el elemento. Una misma clase se puede aplicar a varios elementos diferentes y un único elemento puede tener asignadas varias clases (se indican separadas por espacios en blanco).

Aunque los dos atributos tienen muchos otros propósitos (sobre todo el atributo id), CSS los emplea principalmente con los selectores para indicar los elementos sobre los que se aplican los diferentes estilos.

En el siguiente ejemplo, las dos listas están formadas por un mismo elemento HTML <ul>, pero sus atributos id las distinguen de forma adecuada:

```
<ul id="menu">
  ...
</ul>

<ul id="enlaces">
  ...
</ul>
```

Una de las principales recomendaciones del diseño de páginas XHTML y hojas de estilos CSS está relacionada con los valores asignados a los atributos id y class. Siempre que sea posible, estos atributos se deben utilizar para mejorar la semántica del documento, es decir, para añadir significado a cada elemento de la página.

Por este motivo, se recomienda que los valores asignados a id y class indiquen la función del elemento y no estén relacionados con su aspecto o su posición:

Valores no recomendados	Valores recomendados
negrita	importante
arial15	titular
verdanaPequena	normal
menuIzquierdo	menuSecundario
letraRoja	error

Elegir el valor adecuado para los atributos id o class es sencillo: si el aspecto de un elemento cambia, el valor de id o class debe seguir siendo adecuado. Por tanto, evita utilizar valores relacionados con su posición (izquierdo, derecho, primero, segundo, superior, etc.), color (textoRojo, subrayadoGrisClaro, etc.) o tipo de letra (verdana10, arial15px, etc.)

Técnicamente, los valores de los atributos id y class deben cumplir las siguientes restricciones:

- Sólo pueden empezar por un guión medio (-), un guión bajo (_) o una letra.
- El resto de caracteres, pueden ser números, guiones medios (-), guiones bajos (_) y letras.
- Los navegadores distinguen entre mayúsculas y minúsculas.
- Aunque es posible utilizar letras como ñ y acentos, no se recomienda hacerlo porque no es seguro que funcione correctamente en todas las versiones de todos los navegadores.

13.7.2 CLASSitis y DIVitis.

Un error común al comenzar a desarrollar páginas con estilos CSS es la utilización excesiva de etiquetas y atributos class.

Ejemplo de divitis y classitis:

```
<div id="menu">
<ul class="menu">
  <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
  <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
  <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
  <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
</ul>
</div>
```

Los selectores de CSS permiten prescindir de la mayoría de etiquetas y atributos id y class. Diseñar páginas con exceso de etiquetas no mejora la semántica del documento y sólo consigue complicar el código HTML.

13.7.3 Estructuración del código CSS.

La posibilidad de incluir todo el código CSS en archivos externos exclusivamente dedicados a contener las reglas CSS, permite ordenar de forma lógica las reglas, mejorando su legibilidad y facilitando su actualización.

Las reglas CSS de las hojas de estilos complejas se suelen agrupar según su funcionalidad y se suelen incluir en el siguiente orden:

- Estilos básicos (<body>, tipo de letra por defecto, márgenes de <ul>, <ol> y <li>, etc.)
- Estilos de la estructura o layout (anchura, altura y posición de la cabecera, pie de página, zonas de contenidos, menús de navegación, etc.)
- Enlaces (estilos normales, estilos :hover, etc.)
- Estilos de cada una de las zonas (elementos de la cabecera, titulares y texto de la zona de contenidos, enlaces, listas e imágenes de las zonas laterales, etc.)

Otra característica común de los mejores sitios web es el uso de comentarios CSS para mejorar la estructura de las hojas de estilos muy largas.

13.7.4 División de los estilos en varios archivos CSS

Normalmente, los estilos de una página compleja se dividen en varios archivos CSS diferentes para hacerlos más manejables. En primer lugar, se suele utilizar un archivo común que contiene todos los estilos básicos de las páginas HTML del sitio web.

Además, si existe alguna sección especial del sitio web que requiera nuevos estilos, se crea un archivo CSS con todos esos estilos. También es habitual preparar una hoja de estilos específica para impresora y otra preparada para los dispositivos móviles.

Una vez creados los archivos CSS, existen dos estrategias para enlazar varios archivos CSS en las páginas HTML:

Si se puede modificar fácilmente la cabecera del documento (por ejemplo porque las páginas se generan dinámicamente) lo habitual es incluir tantos elementos <link> como archivos CSS se enlazan:

```
<head>
  ...
  <link rel="stylesheet" type="text/css"
      href="/css/basico.css" media="screen" />
  <link rel="stylesheet" type="text/css"
      href="/css/seccion.css" media="screen" />
  <link rel="stylesheet" type="text/css"
      href="/css/impresora.css" media="print" />
  <link rel="stylesheet" type="text/css"
      href="/css/movil.css" media="handheld" />
  ...
</head>
```

Si no se puede modificar de forma sencilla la cabecera de los documentos para añadir, eliminar y modificar los archivos CSS que se enlazan, lo habitual es enlazar un único archivo CSS que se encarga de importar todos los demás:

```
<head>
  ...
  <link rel="stylesheet" type="text/css" href="/css/estilos.css" media="all" />
  ...
</head>
```

El contenido del archivo estilos.css debería ser el siguiente para ser equivalente al ejemplo anterior:

```
@import url("basico.css") screen;
@import url("seccion.css") screen;
@import url("impresora.css") print;
@import url("movil.css") handheld;
```

14 RECURSOS ÚTILES

Disponer de una buena colección de recursos realmente útiles es una de las características que diferencian a los diseñadores web profesionales del resto.

En primer lugar, resulta imprescindible instalar en el navegador Firefox varias extensiones relacionadas con el diseño web. Todas estas extensiones facilitan el trabajo, ayudan a descubrir rápidamente la causa de los errores del diseño y en general mejoran notablemente la productividad de los diseñadores.

Además, siempre es necesario disponer de varias galerías de páginas como fuente de inspiración y colecciones de enlaces a buenos recursos relacionados con el diseño web. Por último, también es necesario estar al día de las últimas técnicas y novedades mediante sitios web y blogs dedicados en exclusiva al diseño web.

14.1 Extensiones de Firefox

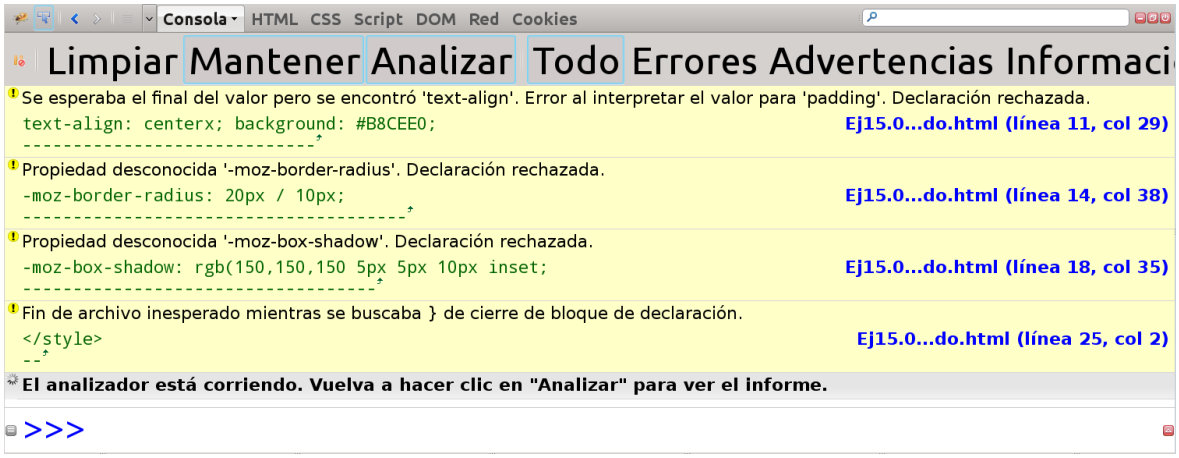
Las principales ventajas de Firefox desde el punto de vista del diseñador y creador de páginas web es que respeta los estándares del W3C mucho más que los navegadores de la familia Internet Explorer. Además, permite instalar pequeños añadidos, llamados extensiones, que añaden funcionalidades al navegador.

Una extensión se puede considerar como un pequeño programa que se instala dentro del navegador y que añade alguna característica interesante que el navegador no incorpora de serie. Lo mejor de las extensiones de Firefox es que existen cientos de extensiones, prácticamente todas son gratuitas y casi todas son realmente útiles.

El sitio web de Firefox incluye una sección especial llamada "Complementos" en la que se puede encontrar el listado completo de extensiones disponibles para Firefox. A continuación se muestra una selección de algunas de las extensiones más interesantes para diseñadores de páginas web.

14.1.1 Firebug.

Firebug es la extensión más útil y completa de todas las que están relacionadas con el diseño web. No importa si tu especialidad es XHTML, CSS, JavaScript, DOM o AJAX, ya que Firebug proporciona toda la información posible sobre cada uno de estos temas.



Como Firebug tiene tantas opciones, sería necesario un libro entero para explicar sus posibilidades. Por ello, lo mejor es que instales la extensión y la pruebes en tus proyectos web.

14.1.2 Web Developer

Antes de que existiera Firebug, la extensión Web Developer era la más útil para los diseñadores web. Se trata de una barra que se instala junto con el resto de herramientas del navegador y que básicamente se puede utilizar para obtener información sobre la página.

Además de proporcionar información, la extensión Web Developer incluye utilidades que Firebug todavía no incorpora, como la redimensión de la ventana del navegador a las dimensiones más utilizadas, recuadrar todos los elementos de un determinado tipo (celdas de tabla, divs, etc.), mostrar una lupa, una regla redimensionable, mostrar los elementos de tipo hidden, mostrar la ruta de cada imagen, etc.

14.1.3 HTML Validator

Una buena práctica, y un requisito impuesto por muchos clientes, es que las páginas XHTML creadas sean válidas y por tanto, pasen el validador de HTML y de CSS disponible en el W3C. Para facilitar la validación de las páginas, la extensión HTML Validator indica en todo momento los errores y las recomendaciones sobre el código HTML de la página que muestra el navegador.

14.1.4 Otras Extensiones

- **ColorZilla**: permite obtener el color de cualquier elemento de la página mediante una herramienta similar a la de los programas de diseño gráfico.
- **MeasureIt**: permite medir la altura y anchura de cualquier elemento de la página.
- **View Source With**: permite elegir el programa o editor con el que se muestra el código fuente de la página y los archivos CSS y JavaScript.
- **Screengrab**: permite guardar una página entera como una imagen. Los pantallazos ("screenshots") también se pueden realizar de una parte concreta de la página o de los contenidos visibles en la ventana del navegador.
- **IE Tab**: permite visualizar con Internet Explorer cualquier página cargada en Firefox. La integración con Internet Explorer es total, ya que ni siquiera hace falta abrir ese navegador.

14.2 Aplicaciones web

A continuación se indican algunas aplicaciones web que pueden ser de utilidad para el diseñador CSS:

- Clean CSS: optimiza, ordena, limpia, corrige y reduce el tamaño de las hojas de estilos.
- Typetester: permite comparar de forma sencilla diferentes tipos de letra y propiedades relacionadas con la tipografía y el texto.
- Browsershots: muestra cómo se visualiza una misma página web en diferentes navegadores de diferentes sistemas operativos (55 navegadores en total). El uso de la aplicación es gratuito y se pueden ver y/o descargar las imágenes que muestran el aspecto de la página en cada navegador.
- Stripe Generator: permite generar fácilmente imágenes preparadas para poder repetirse en todas direcciones de forma correcta y por tanto, para que puedan ser utilizadas como imágenes de fondo.

14.3 Sitios web de inspiración

Muchas veces resulta útil disponer de buenos ejemplos de páginas diseñadas completamente con CSS para tomarlas como referencia y posible inspiración de los diseños propios:

- Web Creme: incluye diariamente varios ejemplos de las mejores páginas diseñadas con CSS y permite realizar búsquedas a partir del color utilizado en la página.
- CSS Remix: muestra centenares de páginas diseñadas exclusivamente con CSS y con la posibilidad de puntuar su diseño.
- CSS Zen Garden: es una galería diferente a las tradicionales, pero se ha convertido en una referencia en cuanto a diseños complejos realizados mediante CSS.
- Open Source Web Design: sitio web que ofrece cientos de plantillas gratuitas con posibilidad de utilizarlas libremente en aplicaciones personales y comerciales.

14.4 Referencias y colecciones de recursos

- Especificación oficial de CSS 2.1: la que incluyen todos los navegadores actuales.
- Especificación oficial de CSS 3: la que sustituirá a la actual versión 2.1.
- del.icio.us/webDesign: colección de recursos relacionados con todos los aspectos del diseño web. Lista creada y mantenida por el diseñador Miguel Sánchez.
- netvibes.com/formacionweb: colección de noticias, artículos y recursos relacionados con el diseño web. Lectura diaria recomendada. Los recursos han sido seleccionados por el diseñador Miguel Sánchez.
- Web Developers Handbook: cientos de enlaces con todos los recursos útiles para diseñadores web.
- Blue Vertigo: otra colección de cientos de enlaces con recursos útiles para diseñadores web.
- Foros del Web: incluyen foros específicos dedicados a resolver dudas relacionadas con el diseño web con CSS.
- Ovillo: una de las mejores listas de distribución en castellano.
- Los foros de SitePoint son una de las mejores referencias en inglés para resolver dudas relacionadas con el diseño web.

15 NOVEDADES EN CSS3 (PARTE I)

15.1 Efectos sobre Cajas y Textos

15.1.1 Border Radius

Con las propiedad border-radius podemos poner las esquinas de la caja de forma redondeada. Veamos un ejemplo: **Ej15.01a-BorderRadius.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Esquinas Redondeadas</title>
  <style type="text/css">

    #principal {  display: block;     width: 500px;
                  margin: 50px auto;  padding: 15px;
                  text-align: center; background: #B8CEE0;
                  border: 2px solid blue;

                  -moz-border-radius: 20px 10px 30px 50px;
                  -webkit-border-radius: 20px 10px 30px 50px;
                  border-radius: 20px 10px 30px 50px;   }

    #titulo {     font: bold 36px verdana, sans-serif;  }

  </style>
</head>
<body>
  <header id="principal">
    <span id="titulo">Estilos CSS 3</span>
  </header>
</body>
</html>
```

La propiedad border-radius en este momento es experimental por lo que debemos usar los prefijos –moz- y -webkit- para que funcionen en navegadores basados en motores Gecko y WebKit, como Firefox, Safari y Google Chrome. Si todas las esquinas tienen la misma curvatura podemos utilizar un solo valor. Sin embargo, como ocurre con las propiedades margin y padding, podemos también declarar un valor diferente por cada una (el funcionamiento es el mismo, siguiendo las agujas del reloj, empezando por la esquina superior izquierda).

También podemos dar forma a las esquinas declarando un segundo grupo de valores separados por una barra. Los valores a la izquierda de la barra representarán el radio horizontal mientras que los valores a la derecha representan el radio vertical. La combinación de estos valores genera una elipsis: **Ej15.01b-BorderElipsis.html**

```
<!DOCTYPE html>
...

    #principal {  display: block;     width: 500px;
                  margin: 50px auto;  padding: 15px;
                  text-align: center; background: #B8CEE0;
                  border: 2px solid blue;

                  -moz-border-radius: 40px / 20px;
                  -webkit-border-radius: 40px / 20px;
                  border-radius: 40px / 20px;  }

    #titulo {     font: bold 36px verdana, sans-serif;  }
  </style>
</head>
<body>
  <header id="principal">
    <span id="titulo">Estilos CSS 3</span>
  </header>
</body>
</html>
```

15.1.2 Sombras

15.1.2.1 La propiedad Box-Shadow

Con la propiedad box-shadow podremos aplicar sombras a nuestras cajas con solo una simple línea de código. La propiedad box-shadow necesita al menos tres valores:

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

NOTA: Está permitido poner varias sombras en box-shadow, separando valores con comas.
Veamos un ejemplo a partir de los anteriores: **Ej15.01c-BoxShadow.html**

```
<!DOCTYPE html>
...
    #principal {
                    ...
                    border-radius: 40px/20px;

                    box-shadow: 10px 5px 10px red,
                                yellow 12px 7px 10px,
                                15px 10px 10px green,
                                25px 20px 5px violet,
                                -10px -5px 10px blue;        }
...
</body>
</html>
```

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

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

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

Agregando otro valor más en pixeles al final de la propiedad desparramará la sombra. Este efecto cambia un poco la naturaleza de la sombra expandiendo el área que cubre.

El último valor posible para box-shadow no es un número sino más bien una palabra clave: inset. Esta palabra clave convierte a la sombra externa en una sombra interna, lo cual provee un efecto de profundidad al elemento afectado. Modificamos el ejemplo: **Ej15.01d-ShadowDifuso.html**

```
<!DOCTYPE html>
...
    #principal {
                    ...
                    box-shadow: rgb(150,150,150) 5px 5px 10px inset;  }
...
</body>
</html>
```

15.1.2.2 La Propiedad Text-Shadow

La propiedad box-shadow fue diseñada especialmente para ser aplicada en cajas. Si intenta aplicar este efecto a un elemento , por ejemplo, la caja invisible ocupada por este elemento en la pantalla tendrá una sombra, pero no el contenido del elemento. Para crear sombras para figuras irregulares como textos, existe una propiedad especial llamada text-shadow:

Modificamos el ejemplo visto antes: **Ej15.01e-TextShadow.html**

```
<!DOCTYPE html>
...
    #principal {
                    ...
    #titulo {       font: bold 36px verdana, sans-serif;
                    text-shadow: rgb(0,0,150) 3px 3px 5px;   }
...
</body>
</html>
```

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

15.1.3 @font-face

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

1. En primer lugar nos descargamos la fuente, por ejemplo de esta dirección: http://www.fontspace.com/alphabet-and-type-typography-and-graphic/star-trek-future
2. Descomprimimos el ZIP por ejemplo en la subcarpeta imagenes
3. Modificamos unos de los TTF para dejarlo así: Star_Trek_Enterprise_Future.ttf
4. Y ahora el cambio del código, siguiendo el ejemplo anterior:

Ej15.01f-FontFace.html

```
<!DOCTYPE html>
...
    #principal {
                    ...
    #titulo {       font: bold 36px MiNuevaFuente, verdana, sans-serif;
                    text-shadow: rgb(0,0,150) 3px 3px 5px;
    }

    @font-face {    font-family: 'MiNuevaFuente';
                    src: url('imagenes/Star_Trek_Enterprise_Future.ttf');
    }
...
    <span id="titulo">STAR TREK ENTERPRISE</span>
  </header>
</body>
</html>
```

IMPORTANTE: Entre url y el paréntesis NO PUEDE HABER ESPACIOS!!.
NOTA: Como alternativa podemos añadir una fuente desde Google Fonts Mas información: www.ideasfrescas.es/es/blog-paginas-web/buscadores/google/google-fuentes-paginas-web.php

Como puede observarse en el código, la forma de añadirlo es con una regla especial o directiva llamada @font-face donde indicaremos un nombre y un atributo de recurso, src: url (" "). Dicho nombre podrá usarse dentro de la propiedad font como otra fuente mas. Se recomienda, de todos modos, incluir al menos un par de fuentes alternativas. Por otro lado el archivo ttf debe estar el servidor o en el mismo equipo (si no se hace, en Firefox no funcionaria).

La forma de hacerlo con Google Fonts sería la siguiente:

1. Nos vamos http://www.google.com/webfonts
2. Elegimos la fuente y le damos a QuickUse
   Por ejemplo: http://www.google.com/webfonts#QuickUsePlace:quickUse/Family:
3. Copiamos el link que ofrece Google y lo ponemos antes del <style>
4. Por último, para añadirlo, usamos el Font-Family que nos pasan.

Un ejemplo completo:

Ej15.01f-FontFaceEspecial.html

```
<!DOCTYPE html>
<html lang="es">  <!-- Ej15.01f-FontFaceEspecial.html -->
<head>
   <meta charset="UTF-8" />
   <title>Fuentes Importadas</title>
   <link
        href='http://fonts.googleapis.com/css?family=Griffy'
        rel='stylesheet' type='text/css' />

   <style type="text/css">
      #principal {
         display: block;
         width: 500px;
         margin: 50px auto;
         padding: 15px;
         text-align: center;
         background: #B8CEE0;
         border: 2px solid blue;

      /* border-radius: 40px/20px;
         box-shadow: 10px 5px 10px red inset,
                  yellow 12px 7px 10px ,
                  15px 10px 10px green,
                  25px 20px 5px violet,
                  -10px -5px 10px blue;*/
      }

      #titulo {
         font-family: Griffy, FuenteStarTrek, Verdana, sans-serif;
         font-size: 60px;
         font-weight: bold;

      /* text-shadow:   5px 0 red,
                     -5px 0 blue;*/
      }

      /*http://www.google.com/webfonts*/

      @font-face {
         font-family: 'FuenteStarTrek';
         src: url("imagenes/Star_Trek_Enterprise_Future.ttf");
      }
   </style>
</head>
<body>
   <header id="principal">
      <span id="titulo">Estilos CSS3</span>
   </header>
</body>
</html>
```

Por supuesto, se pueden añadir tantas fuentes como se quieran (el mismo google indica que, cuantas mas fuentes se añadan mas tiempo tarda la página en cargar).
Como alternativa podemos poner en el navegador el href del link importado, por ejemplo:
http://fonts.googleapis.com/css?family=Griffy
Y luego añadir la directiva que nos aparece dentro del CSS.

15.1.4 Gradientes o Degradados

15.1.4.1 Lineal

Los gradientes son uno de los efectos más atractivos entre aquellos incorporados en CSS3.
Este efecto era prácticamente imposible de implementar usando técnicas anteriores pero ahora es realmente fácil de hacer usando CSS. Una propiedad background con algunos pocos parámetros es suficiente para convertir su documento en una página web con aspecto profesional.
Veamos un ejemplo siguiendo los anteriores. **Ej15.01g-GradienteLineal.html**

```
<!DOCTYPE html>
...
    #principal {
                    ...
                    background: -webkit-linear-gradient(top, #FFFFFF, #006699);
                    background: -moz-linear-gradient(top, #FFFFFF, #006699);
                    ...
                }
...
  </header>
</body>
</html>
```

Los gradientes son configurados como fondos, por lo que podemos usar las propiedades background o background-image para declararlos.
La sintaxis para los valores declarados en estas propiedades es:
`linear-gradient(posición inicio, color inicial, color final).`

Los atributos de la función linear-gradient() indican el punto de comienzo y los colores usados para crear el gradiente. El primer valor puede ser especificado en pixeles, porcentaje o usando las palabras clave top, bottom, left y right (como hicimos en nuestro ejemplo). El punto de comienzo puede ser reemplazado por un ángulo para declarar una dirección específica del gradiente:
`background: linear-gradient(30deg, #FFFFFF, #006699);`

También podemos declarar los puntos de terminación para cada color:
`background: linear-gradient(top, #FFFFFF 50%, #006699 90%);`

15.1.4.2 Radial o Degradado Circular.

La sintaxis estándar para los gradientes radiales solo difiere en unos pocos aspectos con respecto a la anterior. Debemos usar la función radial-gradient() y un nuevo atributo para la forma. Modifcamos el ejemplo anterior: **Ej15.01h-GradienteRadial.html**

```
<!DOCTYPE html>
...
    #principal {
                    ...
      /* Comentamos los gradientes lineales
      background: -webkit-linear-gradient(top, #FFFFFF, #006699);
      background: -moz-linear-gradient(top, #FFFFFF, #006699); */

      /* Ejemplos diferentes para Chrome y Firefox*/
      background: -webkit-radial-gradient(center, ellpse, #FFFFFF 0%, #006699 200%);
      background: -moz-gradient(center, circle, red 30%, blue 40%, yellow 20%, orange 30%);
                    ...
                }
...
</html>
```

La posición de comienzo es el origen y puede ser declarada en pixeles, porcentaje o una combinación de las palabras clave center, top, bottom, left y right.
Existen dos posibles valores para la forma (circle y ellipse).
La terminación para el color indica el color y la posición donde las transiciones comienzan.
NOTA: tanto para Linear como Radial Gradient se pueden poner varios colores.

15.2 Colores

15.2.1 RGBA

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

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

Veamos en primer lugar un ejemplo general donde podemos jugar con esta nueva propiedad. Por ejemplo, podemos aplicarlo a la web del Aeropuerto. Así, podemos poner el contenedor semitransparente sobre un fondo de textura que le dará un efecto muy atractivo.

Ej15.02a-Aeropuerto.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Esquinas Redondeadas</title>
  <style type="text/css">
    body {
      background-image:
      url("http://www.fondosypantallas.com/wp-content/uploads/2010/10/cielo-de-nubes.jpg");
      background-repeat: no-repeat;
      overflow: hidden;
    }

    #contenedor {
      width: 1280px;
      height: 800px;
      background-color: rgba(0, 195, 198, 0.2);
      margin: 0 auto;
      box-shadow: rgb(150,150,150) 5px 5px 10px;
    }

    img { margin: 50px 40px;
          width: 1200px;
          height: 300px;
          opacity: 0.6;
          box-shadow: rgb(150,150,150) 5px 5px 10px;
          }

    h1 {  text-align: center;
          color: yellow;}

  </style>
</head>
<body>
  <div id="contenedor">
    <div id="cabecera">
      <img
src="http://adcstatic.com/imagenes/destinos/l/alquiler-coches-lanzarote-aeropuerto.jpg"
alt="avion" />
    </div>
    <div id="contenido">
      <h1> Aeropuerto Madrid Barajas</h1>
    </div>
  </div>
</body>
</html>
```

15.2.2 HSLA

Del mismo modo que la función rgba() agrega un valor de opacidad a rgb(), la función hsla() hace lo mismo para la función hsl(). La función hsla() es simplemente un función diferente para generar colores, pero es más intuitiva que rgba(). Algunos diseñadores encontrarán más fácil generar un set de colores personalizado utilizando hsla().
La sintaxis de esta función es:
`hsla(tono, saturación, luminosidad, opacidad).`

Veamos el ejemplo que llevamos al completo (en negrita RGBA y HSLA)
Ej15.02b-RgbaHsla.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Esquinas Redondeadas</title>
  <style type="text/css">
    #principal {  display: block;     width: 500px;
                  margin: 50px auto;  padding: 15px;
                  text-align: center; background: #B8CEE0;
                  border: 2px solid blue;

                  -moz-border-radius: 20px / 10px;
                  -webkit-border-radius: 20px / 10px;
                  border-radius: 20px / 10px;

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

/*          background: -webkit-linear-gradient(top, #FFFFFF, #006699);
            background: -moz-linear-gradient(top, #FFFFFF, #006699);*/

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

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

    @font-face {
                  font-family: 'MiNuevaFuente';
                  src: url('imagenes/Star_Trek_Enterprise_Future.ttf');
    }
 </style>
</head>
<body>
  <header id="principal">
    <span id="titulo">STAR TREK ENTERPRISE</span>
  </header>
</body>
</html>
```

Veamos como se especifica:

- Tono **(120)** representa el color extraído de una rueda imaginaria y es expresado en grados desde 0 a 360. Cerca de 0 y 360 están los colores rojos, cerca de 120 los verdes y cerca de 240 los azules.
- El valor saturación **(100%)** es representado en porcentaje, desde 0% (escala de grises) a 100% (todo color o completamente saturado).
- La luminosidad **(50%)** es también un valor en porcentaje desde 0% (completamente oscuro) a 100% (completamente iluminado). El valor 50% representa luminosidad normal.
- El último valor (a), así como en rgba(), representa la opacidad (**0.5** o 50%)..

15.3 Bordes

15.3.1 Outline

La propiedad outline proviene de CSS2 y ha sido expandida en CSS3 para incluir un valor de desplazamiento. Esta propiedad era usada para crear un segundo borde, y ahora ese borde puede ser mostrado alejado del borde real del elemento.
Modifcamos el ejemplo anterior: **Ej15.03a-Outline.html**

```
<!DOCTYPE html>
...
    #principal {
                ...
            outline: 2px dashed #000099;
            outline-offset: 15px;
            }
...
  </header>
</body>
</html>
```

En el código anterior, agregamos a los estilos originalmente aplicados a la caja de nuestra plantilla un segundo borde de 2 pixeles con un desplazamiento de 15 pixeles. La propiedad outline tiene similares características y usa los mismos parámetros que border. La propiedad outline-offset solo necesita un valor en pixeles.
Esta propiedad, es un shorthand de:
`outline-width, outline-style y outline-color.`

15.3.2 Border-image

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

La propiedad border-image toma una imagen y la utiliza como patrón. De acuerdo a los valores otorgados, la imagen es cortada como un pastel, las partes obtenidas son luego ubicadas alrededor del objeto para construir el borde.
Para empezar descargamos la imagen (mejor, elegir algo grande):
http://www.wpclipart.com/page_frames/crayon_border.png
Modificamos el ejemplo anterior: **Ej15.03b-BorderImage.html**

```
<!DOCTYPE html>
...
    #principal {
                ...
            outline: 2px dashed #000099;
            outline-offset: 15px;

            border: 128px;
            border-image:
            url("crayon_border.png") 128 stretch;
               }
...
  </header>
</body>
</html>
```

Para hacer el trabajo, necesitamos especificar tres atributos:

- El nombre del archivo de la imagen, usando url(“”).
- El tamaño de las piezas que queremos obtener del patrón (no confundir con el tamaño del borde, que va justo antes).
- Y algunas palabras clave para declarar cómo las piezas serán distribuidas alrededor.

Existen tres valores posibles para el último atributo:

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

En nuestro ejemplo utilizamos la propiedad border para definir el tamaño del borde, pero se puede también usar border-with para especificar diferentes tamaños para cada lado del elemento. Lo mismo ocurre con el tamaño de cada pieza, hasta cuatro valores pueden ser declarados para obtener diferentes imágenes de diferentes tamaños desde el patrón.

15.4 Transformaciones y Transiciones.

La propiedad transform puede operar cuatro transformaciones básicas en un elemento: scale (escalar), rotate (rotar), skew (inclinar) y translate (trasladar o mover). Veamos cómo funcionan:
Mas información: http://www.w3schools.com/css3/css3_2dtransforms.asp

15.4.1 Transform: scale

Con la propiedad transform: scale, podemos, por ejemplo, duplicar el tamaño de un elemento. Aplicamos transformación duplicando la escala del elemento.
Modificamos el ejemplo anterior, comentando el último ejercicio : **Ej15.04a-Scale.html**

```
<!DOCTYPE html>
...
    #principal {  ...
            /* Comentamos el ejercicio anterior
            border: 128px;
            border-image:
            url("crayon_border.png") 128 stretch; */
               ... }

   #principal:hover{
         -moz-transform: scale(1.5);
         -webkit-transform: scale(1.5);
         }
...
</body>
</html>
```

La función scale recibe dos parámetros: el valor X para la escala horizontal y el valor Y para la escala vertical. Si se da un solo valor, se da para ambos.

Números enteros y decimales pueden ser declarados para la escala. Esta escala es calculada por medio de una matriz. Los valores entre 0 y 1 reducirán el elemento, un valor de 1 mantendrá las proporciones originales y valores mayores que 1 aumentarán las dimensiones del elemento de manera incremental.
Un efecto atractivo puede ser logrado con esta función otorgando valores negativos:

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

Dos parámetros han sido declarados para cambiar la escala de la caja principal. El primer valor, 1, mantiene la proporción original para la dimensión horizontal de la caja. El segundo valor también mantiene la proporción original, pero invierte el elemento verticalmente para producir el efecto espejo. Existen también otras dos funciones similares a scale pero restringidas a la dimensión horizontal o vertical: scaleX y scaleY. Estas funciones, por supuesto, utilizan un solo parámetro.

15.4.2 Transform: rotate

La función rotate rota el elemento en la dirección de las agujas de un reloj. El valor debe ser especificado en grados usando la unidad “deg”. Si se declara un valor negativo, solo cambiará la dirección de rotación del elemento. El máximo son 180grados (180deg).
Modificamos el ejemplo anterior: **Ej15.04b-Rotate.html**

```
<!DOCTYPE html>
...
    #principal {
                    ...
                    margin-top: 200px;
                    -moz-transform: rotate(30deg);
                    -webkit-transform: rotate(30deg);
         }
...
  </header>
</body>
</html>
```

15.4.3 Transform: skew

Esta función cambia la simetría del elemento en grados y en ambas dimensiones.
(es decir, convierte la caja en trapecio).
Modificamos el ejemplo anterior: **Ej15.04c-Skew.html**

```
<!DOCTYPE html>
...
    #principal {  ...
                    -moz-transform: skew(20deg);
                    -webkit-transform: skew(20deg);
         }
...
  </header>
</body>
</html>
```

La función skew usa dos parámetros, pero a diferencia de otras funciones, cada parámetro de esta función solo afecta una dimensión (los parámetros actúan de forma independiente). En el código anterior, realizamos una operación transform a la caja de la cabecera para inclinarla. Solo declaramos el primer parámetro, por lo que solo la dimensión horizontal de la caja será modificada. Si usáramos los dos parámetros, podríamos alterar ambas dimensiones del objeto:

```
 -moz-transform: skew(30deg,20deg);
 -webkit-transform: skew(30deg,20deg);
```

Como alternativa podemos utilizar funciones diferentes para cada una de ellas: skewX y skewY.

15.4.4 Transform: translate

Esta propiedad es similar a top y left de CSS2. Con la función translate el elemento se mueve o desplaza en la pantalla a una nueva posición.

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

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

NOTA: Es importante reseñar que esta transformación debemos ponerla antes de las anteriores, porque dar lugar a errores en las tranformaciones posteriores.

Volvemos a modificar los ejemplos anteriores (pongo todo el código).

Ej15.04d-Translate.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Translate</title>
  <style type="text/css">
    #principal {
      display: block;     width: 500px;
      margin: 50px auto;  padding: 15px;
      text-align: center; background: #B8CEE0;
      border: 2px solid blue;

      -moz-border-radius: 20px / 10px;
      -webkit-border-radius: 20px / 10px;
      border-radius: 20px / 10px;

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

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

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

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

      /* Hay que comentar el resto de transformaciones (hay que ponerlas TODAS con
      el Shorthand que se verá en el apartado siguiente ...
      -moz-transform: scale(1.5);
      -webkit-transform: scale(1.5);

      margin-top: 200px;
      -moz-transform: rotate(30deg);
      -webkit-transform: rotate(30deg);

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

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

    @font-face {
                  font-family: 'MiNuevaFuente';
                  src: url('imagenes/Star_Trek_Enterprise_Future.ttf');
    }
 </style>
</head>
<body>
  <header id="principal">
    <span id="titulo">STAR TREK ENTERPRISE</span>
  </header>
</body>
</html>
```

15.4.5 El Shorthand Transform

A veces podría resultar útil realizar sobre un elemento varias transformaciones al mismo tiempo. Para obtener una propiedad transform combinada, solo tenemos que separar cada función a aplicar con un espacio:
NOTA: Comentamos los últimos ejercicios de transformación anteriores...

Modificamos el ejemplo anterior: **Ej15.04e-Transform.html**

```
<!DOCTYPE html>
...
    #principal
      {
      ...
      /* -moz-transform: translate(100px);
      -webkit-transform: translate(100px);

      -moz-transform: scale(1.5);
      -webkit-transform: scale(1.5);

      margin-top: 200px;
      -moz-transform: rotate(30deg);
      -webkit-transform: rotate(30deg);

      -moz-transform: skew(20deg);
      -webkit-transform: skew(20deg);*/

      -moz-transform: translateY(100px) rotate(45deg) scaleX(0.3);
      -webkit-transform: translateY(100px) rotate(45deg) scaleX(0.3);
      }
...
  </header>
</body>
</html>
```

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

15.4.6 Transformación Dinámica

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

NOTA: Quitamos las transformaciones previas. **Ej15.04f-Dinamica.html**

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

    #principal:hover {
      -moz-transform: rotate(5deg);
      -webkit-transform: rotate(5deg);
    }
...
  </header>
</body>
</html>
```

15.5 Transiciones

Mediante la propiedad transition podemos realizar una transformación dinámica mucho mas completa y suave. El documento oficial de su especificación está aquí:
http://www.w3.org/TR/2012/WD-css3-transitions-20120403/
En primer lugar veamos un ejemplo modificando el anterior:**Ej15.05a-Transition.html**

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

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

15.5.1 Transition-property

Define las propiedades a las que se aplicará la transición. Si no definimos unas propiedades explícitamente o utilizamos la expresión all todas las propiedades susceptibles de cambiar lo harán. Hay dos palabras comodines: all=todas cambiarán y none=ninguna lo hará.
Si lo que queremos es definir un grupo de propiedades la sintaxis es unas a continuación de otras separadas por comas (,):

```
transition-property: width, border-radius, font-size;
```

15.5.2 Transition-duration.

Indica cuánto tiempo llevará el cambio. Las unidades válidas son las de tiempo, segundos, minutos, milisegundos.... Valores negativos son inválidos, equivalen a 0 (cero) que significa que no se produce transición. Si son varias las propiedades indicadas en T-property se pueden declarar un tiempo para cada una. Un ejemplo (incluyendo el transition delay):

```
transition-property: color, width, background-color, left;
transition-duration: 2s, 3s,;
transition-delay: .5s, -1s, 1s, 1.5s;
```

Se interpretará de la siguiente forma:

```
color 2s .5s,
width 3s -1s,
background-color 4s 2s,
left 3s 1.5s
```

Un ejemplo modificando el ejemplo 15.04F: **Ej15.05b-Duration.html**

```
<!DOCTYPE html>
...
    #titulo {     font: bold 36px MiNuevaFuente, verdana, sans-serif;
                  text-shadow: rgba(0,0,0,0.5) 3px 3px 5px;
                  color: hsla(120, 100%, 50%, 0.5);
                  font-size: 20px;                        }

    #titulo:hover {
                  transition-property: color, font-size;
                  transition-duration: 1s;
                  color: green;
                  font-size: 40px;                        }
...
</body></html>
```

15.5.3 Transition-timing-function

Esta propiedad es la que determina cómo se desarrolla la transición en el tiempo estipulado.
Los valores permitidos son:

- Linear lo reparte uniformemente.
- Ease se producen aceleraciones y desaceleraciones al inicio, intermedio, final o mezcladas:

steps la transición no es lineal sino a saltos. Su sintaxis es:
steps(Nº) donde Nº (un valor numérico) son los saltos que se producen en la transición. Similar a una no-transición con pasos intermedios.
steps(Nº, start) Indica el nº de saltos y que se realiza al inicio de cada fracción de tiempo. En el ejemplo, T-duration son 4s, los pasos 5, por lo tanto hay un salto cada 4/5S y se realiza al inicio de cada fracción de tiempo.
steps(Nº, end) Lo mismo que el anterior, pero la transición ocurre al final de cada fracción de tiempo. Es el valor por defecto para "steps".

Veamos un ejemplo completo: **Ej15.05c-TransitionTiming.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Esquinas Redondeadas</title>
  <style type="text/css">
    div {
      width: 150px;       height: 30px;
      margin: 20px 0;     color: white;
      background: green;
    }

    div:hover {
      background: blue;
      width: 800px;
      transition-property: background, width;
      transition-duration: 5s;
    }

    #no-transition   {  transition-timing-function: no-transition ;}
    #linear          {  transition-timing-function: linear;}
    #ease            {  transition-timing-function: ease;}
    #ease-in         {  transition-timing-function: ease-in ;}
    #ease-out        {  transition-timing-function: ease-out ;}
    #ease-in-out     {  transition-timing-function: ease-in-out;}
    #steps5          {  transition-timing-function: steps(5);}
    #steps-start     {  transition-timing-function: steps(5,start);}
    #steps-end       {  transition-timing-function: steps(5,end);}
 </style>
</head>
<body>
  <div id="no-transition">sin transición</div>
  <div id="linear">linear</div>
  <div id="ease">ease</div>
  <div id="ease-in">ease-in</div>
  <div id="ease-out">ease-out</div>
  <div id="ease-in-out">ease-in-out</div>
  <div id="steps5">steps(5)</div>
  <div id="steps-start">steps(start)</div>
  <div id="steps-end">steps(end)</div>
</body>
</html>
```

NOTA: Este código funciona sólo con Firefox.
Para hacerlo funcionar con chrome hay que añadir a cada propiedad en negrita -webkit-

15.5.4 Transition-delay

El retardo o tiempo que trascurre entre la acción que desencadena la transición y el inicio de la misma. Esto es, el retraso con el que comienza la transición. No se resta del tiempo marcado a T-duration. Admite valores de tiempo negativos, pero con una particularidad:
Veamos un ejemplo completo: **Ej15.05d-TransitionDelay.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Esquinas Redondeadas</title>
  <style type="text/css">
    div {
      width: 150px;       height: 30px;
      margin: 20px 0;     color: white;
      background: green;
    }

    div:hover {
      background: blue;
      width: 800px;
      transition-property: background, width;
      transition-duration: 5s;
    }

    #sindelay   { transition-delay: 0;}
    #delay-1s   { transition-delay: -1s;}
    #delay-2s   { transition-delay: -2s;}
    #delay1s    { transition-delay: 1s;}
    #delay2s    { transition-delay: 2s;}
    #delay3s    { transition-delay: 3s;}
    #delay4s    { transition-delay: 4s;}

 </style>
</head>
<body>
  <div id="sindelay">Sin Delay</div>
  <div id="delay-1s">Delay -1s</div>
  <div id="delay-2s">Delay -2s</div>
  <div id="delay1s">Delay 1s</div>
  <div id="delay2s">Delay 2s</div>
  <div id="delay3s">Delay 3s</div>
  <div id="delay4s">Delay 4s</div>
</body>
</html>
```

El valor negativo en T-delay actúa al contrario, adelanta la transición al punto que le correspondería estar al haber trascurrido dicho tiempo. Si tienes cambio de anchura de 0% al 100% en 4 segundos, el delay -2s fuerza a comenzar de inmediato la transición comenzando en el 50% de la anchura y con el 50% del tiempo de T-duration para completarla.

15.5.5 Shorthand Transition

Estas propiedades también tienen una forma acortada o resumida (Shorthand) para declarar todas o algunas de ellas. La sintaxis es:

```
transition: property duration timing-function delay;
```

Si utilizas la forma acortada para declarar varias propiedades a cambiar y con igual o distintos parámetros en cada una de las transitions, con el orden anterior separados por comas (,) cada conjunto de ellas quedaría mas o menos así:

```
transition: color 3s linear 0s, width 1s ease 1s;
```

15.5.6 Propiedades que admiten transiciones Css

Propiedad	Tipo de Valor
background-color	color
background-position	percentage, length
border-bottom-color	color
border-bottom-width	length
border-color	color
border-left-color	color
border-left-width	length
border-right-color	color
border-right-width	length
border-spacing	length
border-top-color	color
border-top-width	length
border-width	length
bottom	length, percentage
clip	rectangle
color	color
crop	rectangle
font-size	length, percentage
font-weight	number It's not that simple.
grid-*	various
height	length, percentage
left	length, percentage
letter-spacing	length
line-height	number, length, percentage
margin-bottom	length
margin-left	length
margin-right	length
margin-top	length
max-height	length, percentage
max-width	length, percentage
min-height	length, percentage
min-width	length, percentage
opacity	number
outline-color	color
outline-offset	integer
outline-width	length
padding-bottom	length
padding-left	length
padding-right	length
padding-top	length
right	length, percentage
text-indent	length, percentage
text-shadow	shadow
top	length, percentage
vertical-align	keywords, length, percentage
visibility	visibility
width	length, percentage
word-spacing	length, percentage
z-index	integer
zoom	number

CONTENIDO TEMA 16

Tomadas de las webs:
http://css3clickchart.com
http://theproc.es/page/guia_referencia_css3?layout=false
http://vagabundia.blogspot.com/2006/09/indice-htmlcss.html
http://css-infos.net/

Cajas
flexbox → Caja flexible. Evolución de la propiedad float (en español)
box-align → Alineamiento de cajas (excelente articulo en español)
box-sizing → Tamaño de la caja (relacionado con calc ())
calc() → Permite especificar valores por cálculos aritméticos
tabs → pestañas con CSS (en español y otro articulo)
regions → Permite conectar distintas cajas por su contenido.
template-layout → Plantilla de estructura
display: grid → Presentación en Rejilla (ojo, solo Explorer 10!)

Fondos e Imágenes
background-attachment: local → Fondo adaptado a contenido (en español)
background-clip → Especifica el área de pintado del fondo
background-origin → Origen del fondo (en español)
background-size → Tamaño de Fondo (en español)
multiple Backgrounds → Múltiples Fondos por elemento
reflect → Reflejo de imágenes
currentColor → Color actual heredado (ejemplos)
filters → Filtros para imágenes aplicados a elementos de página

Animaciones y Máscaras
marquee → Marquesinas (en español)
keyframe animation → Animaciones tipo Flash
3d transforms → Transformaciones 3D (en español y Microsoft!)
mask → Permite crear máscaras
blending-mode → Modos de mezcla (mas info)
pointer-events → Eventos de puntero (en español y mas en Mozilla)

Texto
hiphens → Guiones para texto (ojo, no confundir con word-wrap) (en Español)
word-wrap → Ruptura de Palabras largas
text-overflow → Desbordamiento de texto (en español)
text-stroke → Borde de Texto (en español)
::selection → Pseudo-elemento, focus sobre una selección de texto
columns → Divide el texto en Columnas
exclusions → Flujo de texto (mas ejemplos)

Valores y directivas
toggle() → Agrupa valores por herencia (mas aquí)
rem → Tamaño de fuente usando root em (en español)
@supports → Estilos condicionados al soporte del navegador (en español y otro mas)
variables → Variables en CSS (en español)

16 NOVEDADES EN CSS3 (PARTE II)

16.1 Introducción

En este tema vamos a repasar el resto de propiedades nuevas de CSS3. Muchas de ellas tienen una implementación experimental y su uso aún no está del todo recomendado. Es mas, algunas de las propiedades que presento aquí están planteadas por el W3C pero aún no han sido incorporadas a ningún navegador.

Recomiendo la lectura de este excelente artículo (aunque algunas propieades aún no están implementadas):
http://ksesocss.blogspot.com/2012/04/css-text-level-3-novedades.html

16.1.1 Fases de una especificación W3C

Las especificaciones oficiales del consorcio W3C sobre CS3 están aquí:
http://www.w3.org/Style/CSS/current-work

Las especificaciones del W3C pasan por los siguientes estados:

WD: Working Draft, fase de borrador. Es un documento del W3C publicado para revisión por la comunidad.

LC: Last Call, última llamada. Es un nivel posterior de maduración del borrador. Se proponen fechas límite para revisiones, identifica dependencias conocidas y solicita revisión de los diferentes grupos de trabajo así como revisión pública.

CR: Candidate Recommendation, candidato a recomendación. Es un documento que el W3C cree que ha sido profundamente revisado y satisface los requisitos exigidos por los grupos de trabajo. W3C publica este documento para recoger experiencias de las implementaciones finales.

PR: Proposed Recommendation, recomendación propuesta. Es un informe técnico sobre el grado de madurez alcanzado tras revisiones anteriores y que se expone para la aprobación final del W3C.

REC: Recommendation, recomendación. Un documento REC es una especificación o conjunto de guias que tras una extensa elaboración consensuada, ha recibido la aprobación de los miembros y dirección del W3C. En definitiva, una recomendación del W3C es equivalente a un estándar de otras organizaciones.

16.1.2 Propiedades no incorporadas: Texto

Relación de propiedades de texto plantedas por el W3C pero aún no incorporadas por ningún navegador (a 20 Febrero 2013):

- hanging-punctuation → www.wctutorials.com/reference/css/properties/hanging-punctuation
- punctuation-trim → www.wctutorials.com/reference/css/properties/punctuation-trim
- text-emphasis → www.wctutorials.com/reference/css/properties/text-emphasis
- text-justify (solo IE) → www.wctutorials.com/reference/css/properties/text-justify
- text-outline → http://www.wctutorials.com/reference/css/properties/text-outline
- text-wrap → http://www.wctutorials.com/reference/css/properties/text-wrap
- white-space-collapse → http://www.cssportal.com/css-properties/white-space-collapse.htm

Por otro lado, a día de hoy, están planteadas varias propiedades para el **Flujo de Texto:**
Especificación W3C → http://dev.w3.org/csswg/css3-exclusions/

- wrap-flow
- wrap-through

Artículo de propuesta de Adobe:
http://www.adobe.com/devnet/html5/articles/css3-regions.html

16.2 Texto

16.2.1 text-align-last

Especifica la alineación de la última línea del texto.
Valores (mas info: https://developer.mozilla.org/es/docs/CSS/text-align-last):
start → Izquierda si la orientación es izquierda-derecha.
end → derecha si la orientación es izquierda-derecha.
Left | right | center | justify → Izquierda, derecha, centrado y justificado.

Ejemplo completo: **Ej16.02a-TextAlignLast.html**

```
<!DOCTYPE html> <html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Text Align Last</title>
   <style type="text/css">
      div {   width: 200px;       height: 100px;    }
      p   {   text-align:  justify;  -moz-text-align-last: right; }
   </style>
</head>
<body>
   <div>
      <p>En este tema vamos a repasar el resto de propiedades nuevas de CSS3.
      Muchas de ellas tienen una implementación experimental y su uso aún no está del todo
      recomendado. Es mas, algunas de las propiedades que presento aquí están planteadas
      por el W3C pero aún no han sido incorporadas a ningún navegador.
      </p>
   </div>
</body>
</html>
```

16.2.2 word-break

Especifica la forma de ruptura de palabras en un texto.
Valores: (mas info https://developer.mozilla.org/es/docs/CSS/word-break)
`normal` → Predeterminado (en función de justificado, etc)
`break-all` → Permite rupturas de texto para textos no CJK (Chino, Japones, Coreano)
`keep-all` → Mantiene palabras unidas para textos CJK

16.2.3 word-wrap

Especifica la forma de romper palabras largas sobre las cajas.
Valores: (mas info: https://developer.mozilla.org/es/docs/CSS/word-wrap)
`normal` → Las palabras se rompen solo ante rupturas especificas.
`break-word` → En caso de desbordamiento, la palabra se rompe.
Ejemplo: **Ej16.02b-WordWrap.html**

```
<!DOCTYPE html>
...
   <style type="text/css">
    div {   width: 100px;       height: 100px;    }
    p   {   text-align:  justify;
            /* Probamos a comentar la línea de abajo...*/
            word-wrap: break-word; }
    </style>
...
   <div>
      <p>En este tema vamos a repasar el resto de propiedades nuevas de CSS3.
      Muchas de ellas tienen una implementación experimental y su uso aún no está del todo
      recomendado. Es mas, algunas de las propiedades que presento aquí están planteadas
      por el W3C pero aún no han sido incorporadas a ningún navegador.
      </p>
   </div>
...
```

16.2.4 Hyphens

Pone guiones en textos justificados. Hay dos alternativas para ponerlos:
`-webkit-hyphens: auto;`
`-moz-hyphens: auto;`
`hyphens: auto;` O bien poner `word-break:hyphenate;`

Posibles valores para el primer caso:
`none` → Las palabras no se pueden romper, aunque tengas marcas de ruptura.
`manual` → Las palabras sólo se rompe en los saltos de línea en los que hay caracteres dentro de la palabra que sugieran oportunidades de salto de línea. Los caracteres pueden ser explícitos o condicionales.
`auto` → Las palabras se pueden dividir en los puntos de corte de apropiadas o bien según lo determinado por caracteres de separación de sílabas dentro de la palabra o, como se haya determinado automáticamente mediante un recurso de partición de palabras del idioma correspondiente. Los caracteres condicionales de separación de sílabas dentro de una palabra, si está presente, tienen prioridad sobre los recursos automáticos para determinar los puntos de separación de sílabas.

Ejemplo: **Ej16.02c-Guiones.html**

```
<!DOCTYPE html>
...
   <style type="text/css">
    div {   width: 100px;       height: 100px;    }
    p   {   text-align:  justify;
            -moz-hyphens: auto;}
    </style>
...
   <div>
      <!-- Texto ejemplos anteriores -->
   </div>
...
```

16.2.5 Text-Overflow

Corta el texto automáticamente, añadiendo una elipsis (…).
Mas información: http://vagabundia.blogspot.com/2011/09/la-propiedad-text-overflow.html
https://developer.mozilla.org/es/docs/CSS/text-overflow
IMPORTANTE: Hay que poner un overflow diferente a visible, sino el efecto no se ve.
Valores:
Clip → El corte se realizar en función del contenedor
Ellipsis → Se pone una elipsis (…) en el corte
<string> → Se pone un carácter personalizado en el corte.

Ejemplo: **Ej16.02d-TextOverflow.html**

```
<!DOCTYPE html>
...
   <style type="text/css">
      div {    width: 300px;                 height: 50px;  }

      p {      text-align: justify;          overflow: hidden;
               white-space: nowrap;          /* Aparece el párrafo en una sola línea*/
               text-overflow: ellipsis;                    }

      p:hover {   white-space: normal;       overflow: visible;    }
    </style>
...
   <div>
      <!-- Texto ejemplos anteriores -->
   </div>
```

16.2.6 Text-Stroke

Esta propiedad, por ahora, sólo lo implementa Chrome. Es similar al Text-Shadow.
Bordea el texto, pudiendo incluir grosor, color de borde y color de relleno.
Mas info: http://css-tricks.com/adding-stroke-to-web-text/

Las propiedades en concreto son:
-webkit-text-fill-color: → Color de relleno
-webkit-text-stroke-width → Grosor del borde
-webkit-text-stroke-color → Color del borde
Y el shorthand para los 2 últimos:
-webkit-text-stroke

Ejemplo: **Ej16.02e-TextStroke.html**

```
<!DOCTYPE html>
...
   <style type="text/css">
    div {   width: 100px;       height: 100px;    }
    h3  {   -webkit-text-fill-color: blue;
            -webkit-text-stroke: 1px red;
            /*color: white;*/
            text-shadow:  3px 3px 0 #000,   -1px -1px 0 #000,
                          1px -1px 0 #000,  -1px 1px 0 #000,
                          1px 1px 0 #000;
         }
    </style>
...
   <div>
      <h3> Novedades en CSS3 (Parte II)</h3>
      <!-- Texto ejemplos anteriores -->
   </div>
```

16.2.7 ::selection

El pseudo-elemento ::selection se aplica al seleccionar un texto.
Esta estandarizado excepto para Firefox, que hay que añadir ::-moz-selection
Como en otros pseudo-elementos, se coloca tras el elemento a aplicar, por ejemplo:
p::selection { /* Forma que tendra el texto seleccionado. */

Ejemplo: **Ej16.02f-TextSelection.html**

```
<!DOCTYPE html>
...
   <style type="text/css">
    div {   width: 100px;       height: 100px;    }
...
    p::-moz-selection { color: gold;  background: red; }
    p::selection      { color: gold;  background: red; }
   </style>
...
   <div>
      <h3> Novedades en CSS3 (Parte II)</h3>
      <!-- Texto ejemplos anteriores -->
   </div>
```

16.2.8 Columnas.

Existen diversas maneras de poner columnas mediante CSS. Ya hemos visto el uso de DIVs reconvertidos a celdas de tablas. Ahora vamos a usar otra opción (aún experimental) mas sencilla e intuitiva, que además presenta muchas posibilidades.

Las propiedades serán (para el estándar)
column-count → Número de columnas
column-gap: → Separación entre columnas (distancia)
column-rule: → Raya entre columnas (funciona como border)
IMPORTANTE: Hay que incluir los sufijos de -moz- para Firefox y -webkit- para Chrome.

En el caso de Chrome y Safari existe un grupo de propiedades adicionales muy interesantes:
`-webkit-column-break-after`
`-webkit-column-break-before`
`-webkit-column-break-inside`
Que, resumiendo, añaden un salto de columna según sea el caso.
Mas información: http://www.w3.org/TR/css3-multicol/
https://developer.mozilla.org/en-US/docs/CSS/column-count
Veamos un ejemplo completo: **`Ej16.02g-Columnas.html`**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Columnas</title>
   <style type="text/css">
      div { -moz-column-count: 3;
            -moz-column-gap: 2em;
            -moz-column-rule: 1px solid #ccf;
            -webkit-column-count: 3;
            -webkit-column-gap: 2em;
            -webkit-column-rule: 1px solid #ccf;           }

      h3 {  -webkit-column-break-before: right;            }

      .salto   {  -webkit-column-break-before: right;      }
      .c1      {  margin: 0;          }
   </style>
</head>
<body>
   <div>
      <h3 class="c1"> Columna1</h3> <br />
      <span>   En este tema vamos a repasar el resto de propiedades nuevas de CSS3.
         Muchas de ellas tienen una implementación experimental y su uso aún
         no está del todo recomendado. Es mas, algunas de las propiedades
         que presento aquí están planteadas por el W3C pero aún no han sido
         incorporadas a ningún navegador.
      </span>
      <h3 class="salto"> Columna2</h3>
      <span>   En este tema vamos a repasar el resto de propiedades nuevas de CSS3.
         Muchas de ellas tienen una implementación experimental y su uso aún
         no está del todo recomendado. Es mas, algunas de las propiedades
         que presento aquí están planteadas por el W3C pero aún no han sido
         incorporadas a ningún navegador.
      </span>
      <h3 class="salto"> Columna3</h3>
      <span>   En este tema vamos a repasar el resto de propiedades nuevas de CSS3.
         Muchas de ellas tienen una implementación experimental y su uso aún
         no está del todo recomendado. Es mas, algunas de las propiedades
         que presento aquí están planteadas por el W3C pero aún no han sido
         incorporadas a ningún navegador.
      </span>
   </div>
</body>
</html>
```

16.2.9 font-size-adjust

La propiedad font-size-adjust también establece el tamaño de la fuente; en realidad, el tamaño y la legibilidad de una fuente depende menos del valor dado a font-size que de la proporción entre su ancho y su alto. Esa relación (tamaño de la fuente dividido por la altura de la letra x), es llamada x-height. Cuanto más grande sea ese valor menos legible será la fuente si se trata de fuentes pequeñas. **NOTA:** No se ve el efecto en Chrome; probar en Firefox.

Valores permitidos: none (es decir, el font-size), número (factor multiplicador).
Veamos un ejemplo completo: **Ej16.02h-FontSizeAdjust.html**

```
<!DOCTYPE html> <html lang="es">
<head>
  <meta charset="UTF-8" />  <title>Texto</title>
   <style type="text/css">
      body {   margin: 0;            padding: 0;           /* Reseteo básico de CSS*/
               border: none;         font-size: 8px;       /* Reseteo básico de CSS*/
               font-family: Arial, sans-serif;      }
      h1 { font-size-adjust: 2.5;}
      h2 { font-size-adjust: 2;}
      h3 { font-size-adjust: 1.5;}
   </style>
</head>
<body>
   <div>
      <h1> Titulo Principal</h1>
      <h2> Apartado Principal </h2>
      <h3> SubApartado Principal</h3>
   </div>
</body> </html>
```

16.2.10 font-stretch

La propiedad 'font-stretch' selecciona un tipo normal, condensado o expandido de una familia de fuentes. Los valores de las palabras clave absolutas tienen el siguiente orden, del más estrecho al más ancho:
ultra-condensed | extra-condensed | condensed | semi-condensed
normal | semi-expanded | expanded | extra-expanded | ultra-expanded

NOTA IMPORTANTE: A día de hoy, ningún navegador lo implementa (el mas cercano es Firefox).
Mas información: https://developer.mozilla.org/es/docs/CSS/font-stretch
http://www.w3schools.com/cssref/css3_pr_font-stretch.asp
La palabra clave relativa 'wider' modifica el valor al siguiente más expandido por sobre el valor heredado (mientras no lo aumente por sobre 'ultra-expanded'); la palabra clave relativa 'narrower' modifica el valor al siguiente más condensado por debajo del valor heredado (mientras no lo disminuya por debajo de 'ultra-condensed').
Veamos un ejemplo completo: **Ej16.02i-FontStretch.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>FontStretch</title>
  <style type="text/css">
  body  {   font-size: 30px; }
  h1 {      font-stretch: condensed; }
  p  {      font-stretch: ultra-expanded; }
 </style>
</head>
<body>
    <h1>Título Principal</h1>
    <p> Párrafo con Tamaño de texto predeterminado</p>
</body>
</html>
```

16.3 Animaciones y máscaras

16.3.1 Marquesinas

Antes de nada es importante reseñar que NO DEBE USARSE la etiqueta <marquee>.
Una vez que queda claro esto, veamos las propiedades especificas de CSS para crear y configurar marquesinas (Mas info: http://www.w3.org/TR/css3-marquee/):

- overflow-style → Desbordamiento marquesina (marquee-line | marquee-block).
- marquee-style → Estilo de aparición de la marquesina
  (scroll -de fuera hacia dentro- | slide (se para dentro) | alternate (ambos)
- marquee-repetition → Veces que se repite la marquesina
- -webkit-marquee-increment → Distancia que se mueve en cada pasada.
  (small | medium | large | <unidad, ej: 2px>)
- -webkit-marquee-speed → Velocidad de la marquesina.
  (slow | normal | fast | <distance> | time)

Veamos un ejemplo completo: **Ej16.03a-Marquee.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Esquinas Redondeadas</title>
  <style type="text/css">
    div { width: 400px; height: 300px;}
    p {  text-shadow: 5px 5px 5px blue;

         overflow: -webkit-marquee;
         -webkit-marquee-direction: up;
         -webkit-marquee-style: slide;
         -webkit-marquee-speed: fast;
         -webkit-marquee-repetition: 2;
         -webkit-marquee-increment: 0.5%;
      }
 </style>
</head>
<body>
    <div>
    <p> Párrafo con Tamaño de texto predeterminado</p>
    </div>
</body>
</html>
```

Lecturas Recomendadas:
http://webdesign.about.com/od/csstutorial/a/css-overflow.htm
http://webdesign.about.com/od/css3tutorials/a/marquee-in-css.htm
http://www.gesteves.com/experiments/starwars.html

16.3.2 Animaciones

Las animaciones son una característica nueva de CSS3 que permite realizar este tipo de efectos sin necesidad de plugins adicionales (léase Flash y derivados).
Mas información (en Español): https://developer.mozilla.org/es/docs/CSS/animation
http://www.w3schools.com/css3/css3_animations.asp
Las propiedades a emplear serán:
@keyframes → Directiva donde diseñaremos la animación
animation → El shorthand para animaciones que no incluye animation-play-state property
animation-name → Indica el nombre de la animación creada en el @keyframe
animation-duration → Duración en segundos de la animación. Por defecto 0.
animation-timing-function → Forma de realizar la animación. Por defecto "ease".
Animation-delay → Retraso en el inicio de la animación. Por defecto 0.
animation-iteration-count → Veces que se ejecuta (<num> | infinite). Por defecto 1 .
animation-direction → Dirección animación (normal | alternate | reverse). Por defecto "normal".
animation-play-state → Estado de la reproducción (running | paused). Por defecto "running"

Veamos un ejemplo completo: **Ej16.03b-Animaciones.html**

```
<!DOCTYPE html>
<html lang="es">  <!-- Ej16.03b-Animaciones.html -->
<head>
   <meta charset="UTF-8" />
   <title>Animaciones</title>
   <style type="text/css">
   #animacion
   {     margin: 50px;        width:60px;
         height:40px;         background:#92B901;
         color:#ffffff;               position:relative;
         font-weight:bold;    font-size:20px;
         padding:10px;        border-radius:5px;

         -moz-animation-name: animacion;
         -moz-animation-duration: 5s;
         -moz-animation-timing-function: linear;
         -moz-animation-delay: 2s;
         -moz-animation-iteration-count: infinite;
         -moz-animation-direction: alternate;
         -moz-animation-play-state: running;
   }

   @-moz-keyframes animacion
   {
      0%   {-moz-transform: rotate(0deg);left:0px;}
      25%  {-moz-transform: rotate(20deg);left:0px;}
      50%  {-moz-transform: rotate(0deg);left:800px;}
      55%  {-moz-transform: rotate(0deg);left:800px;}
      70%  {-moz-transform: rotate(0deg);left:800px;background:#1ec7e6;}
      100% {-moz-transform: rotate(-360deg);left:0px;}
   }
   </style>
</head>
<body>
   <div id="animacion">
      CSS3<br>
      <span style="font-size:10px;">Animación</span>
   </div>
</body>
</html>
```

Si queremos hacerlo dinámicamente, podemos configurarlo para que la animación inicie en modo pausado: **-moz-animation-play-state: paused;** y luego al hacer :hover volver al **running**.

16.3.3 Transformaciones 3D

En primer lugar, vayamos al grano y veamos un ejemplo de un cubo:
Veamos un ejemplo completo: **Ej16.03c-Transformacion3D.html**

```
<!DOCTYPE html>
<html lang="es">     <!--Ej16.03c-Transformacion3D.html     -->
<head>
  <meta charset="UTF-8" />
  <title>Cubo 3D</title>
   <style>
      * {/* reseteo básico de CSS */
                 margin: 0;                 padding: 0;
                 border: none;                                          }

      .contenedor {  width: 200px;          height: 200px;
                 margin: 75px 0 0 75px;                          }

      .cubo {    width: 100%;               height: 100%;
                 -moz-perspective: 300px;
                 -moz-transform-style: preserve-3d;
                 -moz-perspective-origin: 150% 150%;      }

      .medio, .frontal, .trasera,
      .derecha, .izquierda, .arriba, .abajo {
                 display: block;            position: absolute;
                 height: 100px;             width: 100px;
                 color: white;              text-align: center;
                 font-size: 60px;           line-height: 100px;
                 font-family: Arial;                             }

      .medio {   border: 5px dashed black;  width: 90px;
                 height: 90px;              background: transparent;      }

      .frontal {  background: rgba(0,0,0,0.3);
                 -moz-transform: translateZ(50px);                      }

      .trasera {  background: green;        color: black;
                 -moz-transform: translateZ(-50px);                     }

      .derecha {  background: rgba(255,0,0,0.7);
                 -moz-transform: translateX(50px) rotateY(90deg) ;      }

      .izquierda{ background: rgba(0,0,255,0.7);
                 -moz-transform: translateX(-50px) rotateY(-90deg) ;    }

      .arriba  {  background: rgba(255,255,0,0.7);
                 -moz-transform: translateY(-50px) rotateX(90deg);      }

      .abajo {      background: rgba(255,0,255,0.7);
                 -moz-transform: translateY(50px) rotateX(-90deg);      }

   </style>
</head>

<body>
   <div class="contenedor">
      <div class="cubo">
         <div class="medio"></div>
         <div class="frontal">1</div>
         <div class="trasera">2</div>
         <div class="derecha">3</div>
         <div class="izquierda">4</div>
         <div class="arriba">5</div>
         <div class="abajo">6</div>
      </div>
   </div>
</body>
</html>
```

Sobre el código anterior trabajaremos para ir viendo las distintas propiedades de Transformaciones 3D de CSS.

Propiedad -moz-perspective (estándar perspective):
El primer elemento a definir es la perspectiva. La perspectiva es lo que nos da la sensación de 3D. Cuanto más lejanos estén los elementos del espectador, más pequeños serán.
Definimos como de rápido se encogen con la propiedad perspective. Cuánto más pequeño sea el valor, más profunda será la perspectiva.

Veamos como cambia la Perspectiva:
Vamos a cambiar la siguiente línea:
-moz-perspective: 300px;
Por esta opciones:
-moz-perspective: 0px;
-moz-perspective: 150px;
-moz-perspective: 600px;
-moz-perspective: 300px;

Propiedad -moz-perspective-origin (estándar perspective-origin):
El segundo elemento a configurar es la posición del espectador, con la propiedad perspective-origin. Por defecto, la perspectiva está centrada en el espectador, pero no siempre es lo adecuado.
Veamos como cambia la Perspectiva: https://developer.mozilla.org/es/docs/CSS/perspective-origin
Permite 2 valores, por ejemplo 100px 100px.
El punto frontal será el tamaño del objeto / 2, es decir 50px 50px.
El primer valor será equivalente a top y el segundo valor equivalente a left.

Propiedad -moz-backface-visibility (estándar backface-visibility):
Con esta propiedad podemos quitar la visualización de la cara posterior.
Posibles valores: visible | hidden (oculto)
Lo mejor es verlo con un ejemplo (siguiendo con el código anterior)
Por ejemplo, en frontal:

```
.frontal {
  background: rgba(   0, 0, 0, 0.3 );
  -moz-transform: translateZ( 50px );
      -moz-backface-visibility: hidden;
}
```

De todos modos siempre podemos quitar el alfa:
background: grey;

Propiedad -moz-tranform-origin (estándar tranform-origin)
Permite cambiar el origen de la transformación.
Queda muy bien explicado en este artículo:
https://developer.mozilla.org/es/docs/CSS/transform-origin

Propiedad -moz-transform-style (estándar transform-style)
Define si los elementos hijos son posicionados en el espacio en 3D o en 2D.
Los valores son flat | preserve-3d.
Evidentemente, usamos la opción **preserve-3d**.
De nuevo, queda muy bien explicado aquí:
https://developer.mozilla.org/es/docs/CSS/transform-style
Mas ejemplos: http://www.netmagazine.com/features/20-stunning-examples-css-3d-transforms

Antes de seguir, veamos un ejemplo de animación aplicado al cubo anterior:

Ej16.03c-EjercicioCubo3D.html

```
<!DOCTYPE html>
<html lang="es">     <!--Ej16.03c-EjercicioCubo3D.html      -->
<head>
   <meta charset="UTF-8"/>
   <title>Cubo 3D</title>
   <style type="text/css">

      * {         margin: 0;                padding: 0;
                  border: none;                          }

      .contenedor {  width: 900px;          height: 900px;
                  margin:  100px auto;                   }

      .cubo {     width: 100%;              height: 100%;
                  -moz-perspective: 0px;
                  -moz-transform-style:preserve-3d;
                  -moz-transform-origin:0px 0px;
                  -moz-animation: cubo 10s linear infinite;
                  -moz-animation-play-state: running;    }

      @-moz-keyframes cubo {
0% {-moz-transform: translateX (0px) translateY(0px) rotateY(0deg) rotateY(0deg);        }
25%{-moz-transform:translateX(-450px) translateY(450px) rotatex(90deg) rotateY(-180deg); }
50%{-moz-transform:translateX(0px) translateY(900px) rotatex(-90deg) rotateY(-180deg);   }
75%{-moz-transform:translateX(450px) translateY(450px) rotatex(-180deg) rotateY(90deg);  }
100%{ -moz-transform:translateX(0px) translateY(0px) rotatex(90deg) rotateY(0deg);       }
      }

      .frontal, .trasera, .derecha,
      .izquierda, .arriba, .abajo {
                  display: block;           position: absolute;
                  height: 100px;            width: 100px;
                  color: white;             text-align: center;
                  font-size: 60px;          line-height: 100px;
                  font-family: Arial;                                  }

      .frontal {  background: blue;         -moz-transform: translateZ(50px);    }
      .trasera {  background: violet;               -moz-transform: translateZ(-50px);   }

      .derecha {  background: green;
                  -moz-transform: translateX(50px) rotateY(90deg);               }

      .izquierda {background: aquamarine;
                  -moz-transform: translateX(-50px) rotateY(90deg);              }

      .arriba {   background: red;
                  -moz-transform: translateY(-50px) rotateX(90deg);              }

      .abajo {    background: orange;
                  -moz-transform: translateY(50px) rotateX(-90deg);              }
   </style>
</head>
<body>
   <div class="ref">
   <div class="contenedor">
      <div class="cubo">
         <div class="medio"></div>
         <div class="frontal">1</div>
         <div class="trasera">2</div>
         <div class="derecha">3</div>
         <div class="izquierda">4</div>
         <div class="arriba">5</div>
         <div class="abajo">6</div>
      </div>
   </div>
</body>
</html>
```

Propiedad Matrix (Matriz)
transform: matrix(a, c, b, d, tx, ty)

```
/* donde  a, b, c, d son los puntos de referencia de la matriz
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
   y tx, ty son las variables de traslado.  */
```

Siendo el valor original:
transform: matrix(1, 0, 0, 1, 0, 0);
Veamos un ejemplo completo: **Ej16.03d-TransformacionMatrix.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Transformacion 3D</title>
  <style type="text/css">
    div {      margin-top: 100px;
               width: 600px;
               height: 400px;
               background: aquamarine;
    }

    #codigo1 { background: gold;
               width: 400px;
               transform: matrix(1, 0, 0, 1, 0, 0);
    }

    #codigo2 {  background: wheat;
               -moz-transform: matrix(1, 0, 0.6, 1, 15em, 0);
               -webkit-transform: matrix(1, 0, 0.6, 1, 250, 0);
               -o-transform: matrix(1, 0, 0.6, 1, 250, 0);
               transform: matrix(1, 0, 0.6, 1, 250, 0);
    }
 </style>
</head>
<body>
  <div>
    <pre id="codigo1">
      Primer Código:

      background: gold;
      -webkit-transform: matrix(1, -0.2, 0, 1, 0, 0);
          -o-transform: matrix(1, -0.2, 0, 1, 0, 0);
             transform: matrix(1, -0.2, 0, 1, 0, 0);
    </pre>
    <pre id="codigo2">
      Segundo Código:

      background: wheat;
     -webkit-transform: matrix(1, 0, 0.6, 1,  250, 0);
          -o-transform: matrix(1, 0, 0.6, 1,  250, 0);
             transform: matrix(1, 0, 0.6, 1,  250, 0);
    </pre>
</div>
</body>
</html>
```

Mas información:
http://msdn.microsoft.com/es-es/library/ie/jj665791%28v=vs.85%29.aspx
http://www.netmagazine.com/features/20-stunning-examples-css-3d-transforms
http://www.digitalicon.es/ejemplos/webkit-transform/webkit-transform.htm

16.3.4 Máscaras

Podemos aplicar máscaras a imágenes y a textos. Las propiedades serán:
Las propiedades serán: -webkit-mask-image y -webkit-background-clip (sólo Chrome)
Imágenes a usar: orchid.jpg, star.png y star.svg que están en http://learning-html5.info/images/

Antes de nada, recomiendo la lectura de la especificación del consorcio:
https://dvcs.w3.org/hg/FXTF/raw-file/tip/masking/index.html
Y la web de MDN: https://developer.mozilla.org/en-US/docs/CSS/-webkit-mask
Excelente ejemplos: http://www.the-art-of-web.com/css/radial-gradients/#.USc38H0VzVM

En primer lugar veamos el código **(NOTA: el SVG no me funciona!)**

Ej16.03e-Mascara.html

```
<!DOCTYPE html>
<html lang="es">     <!-- Ej16.03e-Mascara.html     -->
<head>               <!--http://learning-html5.info/images/ -->
   <meta charset="UTF-8"/>
   <title>Máscara</title>
   <style type="text/css">   /*orchid.jpg y star.png*/
      #mascara {
         -webkit-mask-image: url("imagenes/star.png");     }

      #textomascara {
         font-size: 100px;
         font-weight: bold;
         background: url("imagenes/orchid.jpg");
         -webkit-background-clip: text;
         color: transparent;
      }

      #gradientelineal {
         -webkit-mask-image:
            -webkit-gradient(linear,
                        left bottom, left top,
                        from(rgba(0,0,0,0)),to(rgba(0,0,0,1)));
      }

      #gradienteradial {
         -webkit-mask-image:
         -webkit-gradient(radial,
                     50% 50%, 30, 50% 50%, 60,
                     from(#fff), to(rgba(0,0,0,0)));
      }
   </style>
</head>
<body>
   <h1> Ejemplos de Máscaras</h1>
   <img id="mascara" src="imagenes/orchid.jpg" alt="orquidea" />
   <p> Este es el ejemplo mas sencillo de aplicación de máscara...</p>

   <div id="textomascara"> Mi Texto con Máscara aplicada</div>
   <p> En este ejemplo le aplicamos una máscara a un texto.</p>

   <img id="gradientelineal" src="imagenes/orchid.jpg" alt="orquidea" />
   <p> En este ejemplo aplicamos un Gradiente Lineal.</p>

   <img id="gradienteradial" src="imagenes/orchid.jpg" alt="orquidea" />
   <p> En este ejemplo aplicamos un Gradiente Radial.</p>
</body>
</html>
```

16.3.5 Blend-Mode.

NOTA IMPORTANTE: El Blend-Mode aún está experimental. No funciona en ningún navegador.
Lecturas recomendadas:
http://activ.com.mx/introduccion-a-css-blending/
http://blogs.adobe.com/webplatform/2012/04/04/bringing-blending-to-the-web/
https://dvcs.w3.org/hg/FXTF/rawfile/tip/compositing/index.html

Para probar los distintos modos, en primer lugar debemos usar un fondo tipo textura y una imagen con fondo transparente. Por ejemplo:
http://www.chat-4-free.co.uk/images/image12119637.jpg
http://www.dezignwithaz.com/images/dj-wall-tattoos.png

La propiedad se llama blend-mode.
Los distintos modos son:
normal | plus | multiply | screen | overlay | darken
lighten | color-dodge | color-burn | hard-light | soft-light
difference | exclusion | hue | saturation | color | luminosity

Veamos un ejemplo de implementación sencillo (para un futuro ;D) **Ej16.03f-BlendMode.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Blend Mode</title>
   <style type="text/css">
      div {
         width: 600px;
         height: 400px;
         background: url("imagenes/gradient.jpg");
      }

      img {
         blend-mode: difference;
      }
   </style>
</head>
<body>
   <div>
      <img alt="dj" src="imagenes/dj.png"/> </div>
</body>
</html>
```

16.3.6 Eventos de Puntero

En primer lugar algunas lecturas recomendadas:
http://www.w3.org/TR/SVG11/interact.html#PointerEventsProperty
http://www.csslab.cl/2012/07/27/pointer-events-en-css3/
https://developer.mozilla.org/es/docs/CSS/pointer-events

La propiedad es pointer-events y sus posibles valores son:
auto | none | visiblePainted | visibleFill | visibleStroke | visible | painted | fill | stroke | all | inherit
El valor por defecto es auto.
Veamos un ejemplo: **Ej16.03g-PointerEvents.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>PointerEvents</title>
   <style type="text/css">
      .caja {
         margin: 5%;
         float: left;
         background: grey;
         position: relative;
         width: 35%;
         height: 200px;
         border-radius: 20px;
         box-shadow: 15px 0 0 black;
      }

      .capaencima {
         position: absolute;
         top: 0;
         left: 0;
         width: 100%;
         height: 100%;
         background: rgba(255,0,0,0.2);
      }

      #con .capaencima {
         pointer-events: none;
      }
   </style>
</head>
<body>
   <div class="caja" id="sin">
      <p> Este DIV tiene una capa encima que no permite seleccionar texto
      ni pulsar un enlace.</p>
      <p> Por ejemplo tendremos este enlace
      <a href="http://www.google.es">Página Principal de Google</a>.</p>
      <div class="capaencima"></div>
   </div>
   <div class="caja" id="con">
      <p> Este DIV tiene una capa encima que no permite seleccionar texto
      ni pulsar un enlace.</p>
      <p> Por ejemplo tendremos este enlace
      <a href="http://www.google.es">Página Principal de Google</a>.</p>
      <div class="capaencima"></div>
   </div>
</body>
</html>
```

16.4 Fondos e Imágenes

16.4.1 Múltiples Fondos.

La forma de hacerlo es como anteriormente pero separando los fondos por comas.
Imágenes a tomar: http://www.css3.info/wp-content/uploads/2007/09/body-top.gif
http://www.css3.info/wp-content/uploads/2007/09/banner_fresco.jpg
http://www.css3.info/wp-content/uploads/2007/09/body-bottom.gif

Veamos un ejemplo: **Ej16.04a-MultiplesFondos.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Múltiples Fondos</title>
   <style>
      div { width: 720px;     height: 200px;
            padding: 150px 20px 20px 20px;
            background: url(imagenes/body-top.gif) top left no-repeat,
            url(imagenes/banner_fresco.jpg) 11px 11px no-repeat,
            url(imagenes/body-bottom.gif) bottom left no-repeat;          }
   </style>
</head>
<body>
   <div> Esto es un ejemplo  </div>
</body>
</html>
```

16.4.2 Background-Repeat.

Con CSS-3 se agregan dos nuevos valores **space** y **round** para declarar la repetición horizontal y vertical. Cuando se especifica un sólo valor se entenderá que el segundo es el mismo. Por ejemplo, poner repeat es lo mismo que poner repeat repeat y lo mismo para las otras, a excepción de repeat-x que equivale a repeat no-repeat y repeat-y que es lo mismo que no-repeat repeat.
Sólo Opera, Safari y Chrome permiten este valor múltiple. Sólo Opera trabaja con space y round.
Mas información: https://developer.mozilla.org/en-US/docs/CSS/background-repeat
Vamos a ver las distintas opciones (ojo, probar en Chrome):

- Repeat Repeat → Equivale a repetir X e Y
- Repeat no-repeat → Equivale a sólo repetir X.
- Space → (equivale a **space space**) espacia las imágenes para no cortar ninguna, tanto en horizontal como en vertical. (Solo Opera)
- Round → (equivale a **round round**) rellenará todo el espacio sin cortar las imágenes, escalando imágenes al tamaño libre si fuera necesario.

La imagen a usar será esta:
https://twimg0-a.akamaihd.net/profile_images/2549844400/textura_agua_normal.jpg

Veamos un ejemplo: **Ej16.04b-FondoRepeat.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Fondos</title>
  <style>
    div { width: 800px;   height: 600px;
      background-image: url(imagenes/agua.jpg);
      background-repeat: repeat no-repeat;
      color: white;       font: bold;       }
 </style>
</head>
<body>
 <div>Ejemplo de Contenido</div>
</body> </html>
```

16.4.3 Valor Local.

Con la propiedad background-attachment controlábamos el desplazamiento de la imagen cuando hacemos scroll en la ventana con los valores fixed o scroll. Con CSS-3 tenemos el nuevo valor local. Con el valor fixed la imagen permanece fija en el fondo al desplazar la página. Observe como es recortada por la izquierda (al comparar el ancho con la de los siguientes ejemplos). Esto es debido a que la imagen se fija a la caja de la página, en este caso al borde vertical izquierdo de la página y por tanto no se muestra lo que hay entre ese borde y el borde izquierdo del contenedor.

Veamos un ejemplo: **Ej16.04c-FondoAttachment.html**
La imagen a tomar: http://static.general-play.com/thumbs/d74/84/gp84cf0h1f5i0.jpeg

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Fondos</title>
  <style>
    #articulo {   color:green;
                  border-top:silver dotted 3px;
                  border-bottom:silver dotted 3px;
                  padding-top:0.5em;
                  padding-bottom:0.5em;             }

    .tahmajal {   line-height:2;
                  padding:20px 300px;
                  background-image:url(imagenes/tahmajal.jpeg);
                  background-repeat:repeat-y;
                  background-attachment:fixed;      }
  </style>
</head>
<body>
<! -- Poner el primer párrafo de este apartado... -->
<div id="articulo">
  <div class="tahmajal">
    <h1> Tah Majal</h1>
    <p> <! -- Poner primer párrafo artículo Wikipedia Taj Mahal -->
   </p>
  </div>
</div>
      <p> </p><p> </p><p> </p><p> </p><p> </p>
</body>
</html>
```

Si cambiamos: … **background-attachment:fixed;**
Por: … **background-attachment:scroll;**
El fondo se desplaza conjuntamente con toda la página. Ahora la imagen se fija al propio contenedor y no a la caja de la página como antes .

Con el valor local el desplazamiento se hace respecto a la barra del contenedor si la tuviera, como en este ejemplo con la propiedad {overflow: auto} que recorta lo que sobresale y pone una barra vertical de desplazamiento. En este caso al mover esta barra deberá moverse también el fondo con el contenido. Esto ha funcionado en Opera, Safari y Chrome. Firefox lo ignoran y lo trata como scroll, el valor por defecto.

En el código anterior debemos cambiar la definición del class tajmahal:

```
    .tahmajal {   height:6em;
                  overflow: auto;
                  padding:20px 300px;
                  background-image:url(imagenes/tahmajal.jpeg);
                  background-repeat:repeat-y;
                  background-attachment:local;     }
```

16.4.4 background-clip.

Esta nueva propiedad nos permite situar el fondo recortándolo o mejor dicho, ajustándolo según el borde de la caja del elemento con el valor border-box, de su relleno con padding-box o de su contenido con content-box. Con esta propiedad queda determinada el área de posicionamiento del fondo, es decir, el área donde será dibujado ese fondo. El valor por defecto es border-box, por lo que el fondo se dibuja por defecto en todo el elemento rellenando hasta el borde.

En estos ejemplos tenemos un contenedor donde diferenciamos 4 zonas:

- Una franja azul de grosor 1em que formamos con margin: 1em;
- Con outline: blue solid 1em; damos un contorno al elemento
- Una franja de color rojo con transparencia: border: rgba(255,0,0,0.25) solid 1em;. Es el propio borde del elemento.
- Una franja de color blanco que es el relleno con padding: 1em. El relleno es lo que separa el borde del contenido interior.
- El contenido interior con el texto y con el fondo de la imagen

Por cierto, tomada aquí:
http://www.disfrutandosevilla.com/wp-content/uploads/2009/09/catedraldesevilla.jpg

En este primer ejemplo recortamos el fondo al contenido con content-box. El fondo se ajusta sólo al contenido de texto. Safari no ejecuta esto y lo aplica como el ejemplo border-box.

Ej16.04d-BackGroundClip.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Fondos</title>
  <style>
    #articulo { border-top:silver dotted 3px;
                border-bottom:silver dotted 3px;
                padding-top:0.5em;
                padding-bottom:0.5em;             }

    .Sevilla {  margin: 2em;         outline: blue solid 1em;
                padding: 1em;                border: rgba(255,0,0,0.25) solid 1em;
               background-image:url(imagenes/Sevilla.jpg);
                background-clip: content-box;    }

    p, h1     { color:white;         font-size: 20px;
                padding: 0 10px;                 }

  </style>
</head>
<body>
<div id="articulo">
  <div class="Sevilla">
    <h1> Sevilla</h1>
    <p> <!-- Contenido Wikipedia Sevilla (hasta Indice) --> </p>
  </div>
</div>
</body>
</html>
```

Ahora recortamos el fondo al contenido con padding-box. El fondo se ajusta ocupando el relleno:
background-clip: content-box;
Lo cambiamos por: **background-clip: padding-box;**

Con el valor border-box el fondo también rellena el borde. Como éste es de color rojo transparente y se dibuja encima del fondo, la transparencia roja deja entrever el fondo y produce un color diferente:
background-clip: border-box;

16.4.5 La propiedad background-origin.

Esta propiedad especifica donde empieza a dibujarse el área de posicionamiento del fondo, área que se determina con la propiedad {background-clip}. En definitiva, determina el lugar donde comienza a dibujar el fondo. Con content-box comienza a dibujar en el borde superior izquierdo del contenido. Con padding-box comienza en el borde superior izquierdo del relleno. Con border-box lo hará en la esquina superior izquierda del propio borde. El valor por defecto es padding-box.

La imagen a tomar será esta:
http://www.ubuntu-es.org/files/pictures/picture-104065.png
Ej16.04e-BackGroundOrigin.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Fondos</title>
   <style>
      .d1 {    margin:1px;    outline: blue solid 1px;
            width:400px;      float:left;
            padding:0.7em;    border: 0.7em solid rgba(255,0,0,0.25);
            background-origin:content-box;
            background-repeat:no-repeat;
            background-image:url(imagenes/tux.png);                }
   </style>
</head>
<body>
   <div class="d1">
   <!-- Párrafo del principio de esta misma página ... -->
   </div>
</body>
</html>
```

Para comprobar como cambia, debemos irnos a la propiedad:
`background-origin:content-box;`
Y cambiarlo por:

```
background-origin:padding-box; /*(valor predeterminado) */
background-origin:border-box;
```

Y vemos como se va moviendo el bueno de Tux.

16.4.6 background-size.

Esta propiedad establece el tamaño del fondo. Con el valor contain se escalará la imagen conservando su proporción ancho-alto para que ocupe toda el área de posicionamiento disponible, la cual queda determinada por las propiedades {background-clip} y {background-origin}. Si estas no se especifican toman los valores border-box y padding-box, por lo que el área de posicionamiento del fondo será todo el elemento incluso el borde aunque una imagen deberá quedar posicionada en la esquina superior izquierda del relleno.

Los valores posibles son contain, cover o dos valores de dimensiones de longitud, porcentajes o auto para el ancho y alto. Los porcentajes son relativos a las dimensiones del contenedor. Si se da un único valor de dimensión, porcentaje o auto se aplicará al ancho y alto al mismo tiempo.
Para el ejemplo tomaremos el anterior cambiado: **Ej16.04f-BackGroundSize.html**

```
<!DOCTYPE html>
...
       background-origin:padding-box;
       background-size: 32px auto;
       background-repeat:no-repeat;
       background-image:url(imagenes/tux.png);
```

Cambiamos `background-size: 32px auto;`
Por `background-size: contain;` o bien `background-size: cover;`

16.4.7 La propiedad Box-Reflect.

Con esta propiedad podemos reflejar cualquier elemento. Solo funciona Chrome.
Este caso, lo aplicaremos a una imagen.
La imagen a tomar será esta: http://davidwalsh.name/demo/ricci-url.jpg

Ej16.04g-BoxReflect.html

```
<!DOCTYPE html>
<html lang="es">     <!--  Ej16.04g-BoxReflect.html -->
<head>
   <meta charset="UTF-8"/>
   <title>Reflejos</title>
   <style type="text/css">
      img { width: 300px;     }
      #reflejo {
         -webkit-box-reflect:
            below 0
            -webkit-gradient( linear,
                              right top, left bottom,
                              from(transparent),
                              to(white));
      }
   </style>
</head>
<body>
   <img src="imagenes/ricci-url.jpg"
       alt="Christina Ricci Original" />
   <img src="imagenes/ricci-url.jpg" id="reflejo"
       alt="Christina Ricci Reflejo" />
</body>
</html>
```

Los valores permitidos son:

above | below |right | left	→ Posición del reflejo (arriba, abajo, derecha, izquierda)
<length>	→ Distancia del reflejo.
<image>	→ Para aplicar una máscara al reflejo

16.4.8 CurrentColor.

El valor CurrentColor es nuevo en CSS3. Permite definir un color heredado de un elemento anterior. Lo mejor es verlo con un ejemplo: **Ej16.04h-CurrentColor.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>CurrentColor</title>
  <style>
    .contenedor { color:darkred;}
    .cajalinea  { background:currentcolor;  height:10px;}
  </style>
</head>
<body>
  <div class="contenedor">
      El color de este texto es el mismo que el de la línea inferior:
      <div class="cajalinea"></div>
      Mas texto (sin el fondo anterior)
   </div>
</body>
</html>
```

NOTA IMPORTANTE: La utilidad de este valor es limitada. Si, por ejemplo, ponemos un color diferente en .cajalinea (por ejemplo: color:green) el fondo que tomará será precisamente ese, y no el del elemento "padre".

16.4.9 Filtros

Mediante los filtros se pueden aplicar cambios en imágenes usando únicamente CSS.
Esta tecnología aún es experimental: la recomendación del W3C es del 12 de Febrero de ¡2013!.
https://dvcs.w3.org/hg/FXTF/raw-file/tip/filters/index.html
Mas información: https://developer.mozilla.org/es/docs/CSS/filter

La imagen a tomar será esta:
http://blog.centauro.net/wp-content/uploads/190134416_287b52b0c31.jpg

Los filtros disponibles (solo en últimas versiones de Chrome) son:

- blur(radius) → Desenfoque Gausiano (Radio, 0 ~ 5px)
- brightness(amount) → Brillo (Cantidad) [-1 / 1] (valor negativo oscurece)
- contrast(amount) → Contraste (Cantidad)
- drop-shadow(<shadow>) → Sombra
  - <offset-x> <offset-y> → Desplazamiento X, Desplazamiento Y
  - <blur-radius> → Desenfoque Gaussiano
  - <spread-radius> → Tamaño (radio)
  - <color>
- grayscale(amount) → Escala de Grises (Cantidad) (0-1)
- hue-rotate(angle) → Rotación de Matiz (Ángulo) (0deg-360deg)
- invert(amount) → Invertir (Cantidad) (0-100%)
- saturate(amount) → Saturación (Cantidad) (%)
- sepia(amount) → Sepia (Cantidad) (0-1)
- opacity(amount) → Opacidad(Cantidad) [0-100% (opaco)]

En algunas propiedades, los valores van de 0 (mas cerca del original) al 1 (efecto completo).
En las propiedades invert y contrast, con el valor 50% se ve completamente gris.
En contrast, para ver el efecto hay que poner un valor por encima del 100% (por ej: 300%);
Veamos un ejemplo: **Ej16.04i-Filtros.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8"/>
   <title>Filtros</title>
   <style type="text/css">
      img { float: left;  margin: 0 20px;}
      #f1 { -webkit-filter: grayscale(0.8) blur(1px);}
   </style>
</head>
<body>
   <img src="imagenes/aeropuerto.jpg" alt="Foto Original" />
   <img id="f1" src="imagenes/aeropuerto.jpg" alt="Aeropuerto Sevilla" />
   <img id="f2" src="imagenes/aeropuerto.jpg" alt="Aeropuerto Sevilla" />
</body>
</html>
```

Tenéis un ejemplo en tiempo real en esta página:
http://html5-demos.appspot.com/static/css/filters/index.html

16.5 Valores y Directivas

16.5.1 Directiva @toggle

Mediante esta directiva podemos definir un grupo de valores aplicables a un selector.
NOTA IMPORTANTE: La especificación es del 15 Feb 2013. Ningún navegador lo soporta aún.
http://dev.w3.org/csswg/css3-values/#toggle
Lo mejor es verlo con un ejemplo: **Ej16.05a-Toggle.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8"/>
   <title>Toggle</title>
   <style type="text/css">
      ul    {  list-style-type: disc;}
   ul ul    {  list-style-type: toggle(disc, circle, square, box); }
   </style>
</head>
<body>
   <ul>
      <li>Tema1</li>
      <li>Tema2</li>
      <li>
         <ul>
            <li>Sección A</li>
            <li>Sección B</li>
            <li>
               <ul>
                  <li>Apartado1</li>
                  <li>Apartado1</li>
               </ul>
            </li>
            <li>Sección C</li>
            <li>Sección D</li>
         </ul>
      </li>
      <li>Tema3</li>
   </ul>
</body>
</html>
```

16.5.2 La unidad Rem

Funciona de manera muy similar al Font-Strech visto en el apartado 16.2.9.
Es una unidad que depende directamente del elemento raiz.
Mas información: http://www.w3.org/TR/css3-values/#rem-unit
Un ejemplo:**Ej16.05b-UnidadRem.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8"/>
   <title>Toggle</title>
   <style type="text/css">
      html  {  font-size: 20px;       }
      h1    {  font-size: 1.5rem;     }
      p     {  font-size: 0.6rem;     }
   </style>
</head>
<body>
      <h1>Titulo Ejemplo 150% REM </h1>
      <p>Párrafo de ejemplo, 60% REM </p>
</body>
</html>
```

16.5.3 La directiva @supports

Mediante esta directiva, podemos definir estilos condicionados al soporte del navegador.
Es decir, realizar hacks de manera mas elegante a lo hecho hasta ahora.
La especificación: http://dev.w3.org/csswg/css3-conditional/#at-supports
Mas información: https://developer.mozilla.org/en-US/docs/CSS/@supports
NOTA IMPORTANTE: El único que lo soporta es Firefox.
Nos metemos en la barra de direcciones y escribimos about:config
Hay que poner la directiva a true (doble clic) `layout.css.supports-rule.enabled`

Como siempre, lo mejor es verlo con un ejemplo:
Ej16.05c-Supports.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Supports</title>
   <style type="text/css">
      /* ESTO ES UN HACK : Si el navegador es antiguo aplicar bordes normales*/
      h1 {  border-bottom: 2px solid red;
            border-right: 2px solid red;    }

      .p1 { border-bottom: 5px solid blue;
            border-right: 5px solid blue;   }

      /* Si el navegador es moderno, aplicar BoxShadow*/
      @supports (box-shadow: 1px 1px 1px black)
      or (-moz-box-shadow: 1px 1px 1px black)
      or (-webkit-box-shadow: 1px 1px 1px black)
      {
         h1
         {  border: none;
            -webkit-box-shadow: 2px 2px 2px red;
            -moz-box-shadow: 2px 2px 2px red;
            box-shadow: 2px 2px 2px red;           }

         .p1
         {  border: none;
            -webkit-box-shadow: 5px 5px 5px blue;
            -moz-box-shadow: 5px 5px 5px blue;
            box-shadow: 5px 5px 5px blue;          }
      }
    </style>
  </head>
  <body>
      <h1> Título de Ejemplo</h1>
      <p class="p1">Párrafo de Prueba 1</p>
  </body>
</html>
```

16.5.4 Otras Directivas

Las siguientes directivas están especificadas, pero ya sea por ser muy experimentales o no tener mayor utilidad, no las desarrollaré en este manual (a versión 22 Febrero 2013).
@viewport → Adaptación a dispositivos.
Mas información: http://emiliocobos.net/css-directiva-supports-viewport-document/

@page → Paginación mediante CSS
Mas info: http://ksesocss.blogspot.com/2012/05/estilos-css-para-imprimir-la-regla-page.html

@namespace → Espacios de nombres
Mas info: http://www.posicionamientowebenbuscadores.com/rec-css3-namespace-20110929/

16.5.5 Variables en CSS

La declaración de variables es muy útil porque nos permite modularizar el código.
El estándar es de Abril de 2012: http://www.w3.org/TR/css-variables/

En Chrome podemos comprobar muchas configuraciones avanzadas poniendo en el navegador: chrome://flags/. Debemos activar las variables CSS.
NOTA: He probado en Chrome y Firefox y no ha funciona aún. (26feb 2013).
De todos modos, según la recomendación del W3C el código debería ser como el de abajo:
Veamos un ejemplo:**Ej16.05d-Variables.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Variables</title>
   <style type="text/css">
      ::root {    -webkit-var-blanco: white;
               -webkit-var-rojo: red;
               var-miborde: 2px dashed blue;        }

      .d1 {    color: var(blanco);
               background: var(rojo);
               border: var(miborde);         }
    </style>
  </head>
  <body>
      <div class="d1">  Texto Blanco sobre fondo Rojo </div>
  </body>
</html>
```

16.5.6 La pseudo-clase nth-child

Lo que hace es identificar un elemento hijo dentro de las propiedades del elemento padre.
Además, permite usar valores calculados (por ej: 2n+1 para los elementos impares).
El índice n empieza en 0. Por ejemplo, impares excepto el 1º sería 2n+3.
Mas información: https://developer.mozilla.org/en-US/docs/CSS/:nth-child
El estándar: http://www.w3.org/TR/css3-selectors/
Como siempre, lo mejor es verlo con un ejemplo: **Ej16.05e-NthChild.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8"/>
   <title>PseudoClases</title>
   <style type="text/css">
      ul li {  background: aquamarine;
               color: red;                          }
      ul li:nth-child(2n+1) {
               background-color: pink;
               color: blue;
               font-weight: bold;            }
   </style>
</head>
<body>
   <h3> Salidas</h3>
   <ul>
      <li> Vuelo Sevilla - Madrid </li>
      <li> Vuelo Sevilla - Barcelona </li>
      <li> Vuelo Sevilla - Ginebra </li>
      <li> Vuelo Sevilla - París </li>
      <li> Vuelo Sevilla - Valencia </li>
      <li> Vuelo Sevilla - Bilbao </li>
   </ul>
</body>
</html>
```

16.5.7 Las Pseudo-clases :first-child y :last-child

La pseudo-clase :first-child agrega estilo sólo al primer elemento de cierto tipo, contenido dentro de otro. Por ejemplo, si una lista (UL) contiene varios items (LI), podemos hacer que el primero de ellos se muestre diferente del resto. La pseudo-clase :last-child es justo lo contrario.

Siguiendo con el ejemplo anterior: **Ej16.05f-FirstLastChild.html**

```
<!DOCTYPE html>
...
   <style type="text/css">
   ...
      ul li:first-child, ul li:last-child {
         background-color: pink;
         color: blue;
         font-weight: bold;
      }
...
</html>
```

16.5.8 Las pseudo-clases :first-of-type y :last-of-type.

Estas dos pseudo-clases funcionan de manera muy similar al first-child y last-child.
En este caso solo afecta al primer elemento del tipo definido (y no los demas).
LastType funciona igual solo que para el último tipo de elemento.
Seguimos con ejemplo anterior modificado: **Ej16.05g-FirstLastType.html**

```
<!DOCTYPE html>
...
      ul li:first-of-type {
         background-color: pink;
         color: blue;
         font-weight: bold;   }
   </style>
</head>
<body>
   <h3> Salidas</h3>
   <ul>
      <li> Vuelo Sevilla - Madrid </li>
      <li>
         <ul>
            <li>Salida 08:00</li>
            <li>Salida 10:00</li>
            <li>Salida 12:00</li>
         </ul>
      </li>
      <li> Vuelo Sevilla - Barcelona </li>
...
```

16.5.9 La pseudo-clase :not.

Permite especificar definiciones hacia aquellos elementos que no cumplan una condición.
Veamos un ejemplo a partir de los anteriores: **Ej16.05h-PseudoClaseNot.html**

```
<!DOCTYPE html>
..
               ul li:not(.normal) {
                      background: gold;     }
        </style>
        ...
               <li class="normal"> Vuelo Sevilla - Barcelona </li>
               <li> Vuelo Sevilla - Ginebra </li>
               <li class="normal"> Vuelo Sevilla - París </li>
        ...
</body>
</html>
```

16.6 Flexbox

16.6.1 Propiedad Display Box.

El concepto de Cajas flexibles es revolucionario. Para empezar veremos un ejemplo muy sencillo de Layout con 3 DIVs colocados con el mismo tamaño.
En un momento dado, incluyo una regla para hacer que el DIV central tenga un tamaño específico.
Hay que activar (true, doble clic) en about:config la directiva:
`layout.css.flexbox.enabled`

Antes de seguir, algunos enlaces con mas información:
http://www.html5rocks.com/es/tutorials/flexbox/quick/
http://ksesocss.blogspot.com/2012/11/flexbox-nuevo.html

Como siempre, lo mejor es verlo con un ejemplo:
Ej16.06a-Flexbox.html

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8"/>
   <title>Cajas Flexibles</title>
   <style type="text/css">
      #flexbox {
         width: 900px;
         height: 600px;
         border: 1px solid red;
         font: 14px Arial;

         display: -moz-box;
         -moz-box-orient: horizontal;
      }

      #flexbox div {
         -moz-box-flex: 1;
      }

      #caja1 { background: aquamarine; }
      #caja2 { background: cyan;
         /* Con esta regla, hacemos que el DIV central tenga un tamaño
            y los otros se adapten
               width: 400px; */
         }
      #caja3 { background: violet; }

   </style>
</head>
<body>
   <div id="flexbox">
      <div id="caja1"><p>Caja1</p></div>
      <div id="caja2"><p>Caja2</p></div>
      <div id="caja3"><p>Caja3</p></div>
   </div>
</body>
</html>
```

También podemos hacer que la distribución en vertical con los elementos unos encima de otros:

```
#flexbox {
         width: 900px;
         height: 600px;
         border: 1px solid red;
         font: 14px Arial;

         display: -moz-box;
         -moz-box-orient: vertical;
      }
```

16.6.2 Estructura con Cajas Flexibles

Ahora vamos a realizar un ejemplo diferente usando algunas propiedades nuevas.
Como siempre, lo mejor es verlo con un ejemplo:

Ej16.06b-FlexLayout.html

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8"/>
   <title>Cajas Flexibles</title>
   <style type="text/css">
   #principal {      min-height: 800px;
                     display: -webkit-flex;  /*creamos el flex container:*/

                     /*Cómo se disponen los items:*/
                     webkit-flex-direction: row;
                     -webkit-flex-wrap: wrap;       }

   #principal > article {
                     -webkit-flex-grow: 3;
                     -webkit-flex-shrink: 1;
                     -webkit-flex-basis: 50%;
                     -webkit-order: 2;               /*2= se dibujará en segundo lugar*/
                     background: pink;              }

   #principal > nav {   -webkit-flex: 1 6 20%;
                        -webkit-order: 1;           }

   #principal > aside {    -webkit-flex: 1 6 20%;
                        -webkit-order: 3;           }

   header, footer {     min-height: 100px;
                        background: yellow;         }

   nav, aside {         background: palegreen;
                        margin: 0 2.5%;             }

   </style>
</head>
<body>
   <header><h1>header</h1></header>
   <div id='principal'>
      <article><h1>article</h1></article>
      <nav><h1>nav</h1></nav>
      <aside><h1>aside</h1></aside>
   </div>
   <footer><h1>footer</h1></footer>
</body>
</html>
```

16.6.2.1 Los contenedores flexibles: flex e inline-flex

Veamos parte del código:

```
#principal {   display: flex;
               display: -webkit-flex;}
```

A partir de ese momento el contenedor (#principal) será un contenedor flexible y los elementos dentro del contenedor serán elementos flexibles que se adaptarán a los espacios disponibles que les deje el contenedor. Podríamos haber puesto también:

```
#principal {
    display: inline-flex;
    display: -webkit-inline-flex;
}
```

en cuyo caso se comportaría en relación a otros elementos de la página de manera semejante a display: inline. En caso contrario se comporta igual que display: block.

16.6.2.2 Orientación: flex-flow, flex-direction y flex-wrap

La propiedad flex-flow es el shorthand de flex-direction y flex-wrap.

- **flex-direction:**
  row → se alinean en filas.
  row-reverse → en filas, pero con el orden inverso.
  column → se alinean en columnas.
  column-reverse → en columnas, pero con el orden inverso.
  Especifica cómo se colocan los elementos en el contenedor flexible definiendo el eje principal y la dirección (normal o invertida/reverse).

- **flex-wrap:**
  nowrap → El contenedor consta de una sola línea.
  wrap → El contenedor tiene múltiples líneas.
  wrap-reverse → El contenedor tiene múltiples líneas que se colocan en orden inverso.
  Controla si el contenedor flexible (en este caso #viajes) tiene una sola línea o múltiples líneas, así como la dirección en la que se colocan las nuevas líneas.

16.6.2.3 La propiedad order

Podemos establecer el orden en el que aparecen los componentes de una caja flexible. Por defecto aparecerán tal y como aparecen en el código HTML (equivale a order: 0), pero eso se puede cambiar de forma sencilla.
Basta con poner `-webkit-order: n;`
Siendo n el orden de menor a mayor en la que se renderizará el elemento.

16.6.2.4 La propiedad flex

En el apartado 16.6.1 ya vimos esta propiedad con un ejemplo. Veamos cómo funciona.
Con la propiedad flex podemos establecer cómo crece o decrece un elemento flexible dentro del contenedor en relación a los demás.
Así, podemos hacer lo siguiente:

```
#principal >article {
        flex: 2;       }
#principal >aside, #principal >nav {
        flex: 1        }
```

En este caso definimos que la parte de artículos sea el doble que aside y nav.
Esta propiedad se puede volver más compleja, porque puede tener tres parámetros:
flex-grow, flex-shrink y flex-basis.

- **flex-grow:**
  Especifica el factor de crecimiento, es decir, cuanto crecerá el elemento en relación a los demás cuando hay espacio disponible del contenedor a ocupar.
  Por defecto es '1', que es el valor que dimos en el ejemplo anterior a los tres elementos.
- **flex:-shrink:**
  Determina el factor de reducción, es decir, cuanto decrecerá el elemento en relación a los demás cuando hay espacio negativo en el contenedor (el contenedor es más pequeño de los anchos combinados de los elementos que hay en su interior).
  Por defecto es '1′.
- **flex-basis:**
  Toma el mismo valor que la propiedad 'width' y establece el tamaño inicial del elemento antes de distribuir el espacio libre de acuerdo con los ratios de flex-grow o flex-shrink.
  Cuando se omite, su valor es cero.

Para comprender las propiedades anteriores, lo mejor es poner un ejemplo. Supongamos que tenemos un contenedor al que llamaremos “A” que tiene 300px de ancho. Hacemos que este contenedor sea flexible:

```
A { display: flex;}
```

Supongamos que este contenedor tiene en su interior dos elementos, B y C, que no tienen un ancho especificado. Vamos a establecer los siguientes valores para ambos de la propiedad flex: flex-grow, flex-shrink, flex-basis:

```
B { flex: 3 1 100px;}
C { flex: 1 2 100px;}
```

Como hemos establecido un flex-basis para cada elemento de 100px, nos quedarán aún 100px libres sin ocupar (300px (A) - 100px (B) - 100px (C).
¿Como se reparte ese espacio disponible entre los elementos B y C?
En función del flex-grow:
3 partes para el elemento B (75px) y una parte para el elemento C (25px).
Es decir, de inicio el elemento B ocupará 100px+75px = 175px de los 300px disponibles de A
Y el elemento C ocupará 100px+25px=125px de los 300px disponibles del elemento A.

Ahora bien, supongamos que el elemento contenedor A mide 170px y no 300px.
Eso quiere decir que habrá espacio negativo, porque los elementos B y C tienen un flex-basis de 100px cada uno, es decir, 200px, que es 30px mayor que los 170px del contenedor.
En este caso el ratio que se usa es el flex-shrink, que recordemos que era 1 para el elemento B y 2 para el elemento C.
Esos 30px se restarán del ancho de los elementos B y C en función de dicho ratio:
al elemento B se le quitarán 10px y al elemento C se le quitarán 20px.

16.6.3 Como centrar un elemento con flexbox

Usando box-pack y box-align. Funciona con firefox usando -moz-
Pues la mejor manera es viéndolo con código:

Ej16.06c-FlexCentrado.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Cajas Flexibles</title>
   <style type="text/css">
      #contenedor {  width: 600px;    height: 300px;
                  font-size: 20px;   background: khaki;
                  border: 2px solid red;

                  display: -webkit-box;
                  -webkit-box-orient: horizontal;
                  -webkit-box-pack: center;
                  -webkit-box-align: center;        }

      textarea {     margin: 0;       padding: 0;
                  max-width: 600px;  max-height: 300px;

                  /* Para hacerlo fijo:
                  resize: none; */                  }
   </style>
  </head>
  <body>
   <div id="contenedor">
      <textarea> Esto es un TextArea que permite cambios en tiempo real</textarea>
   </div>
  </body>
</html>
```

16.6.4 La propiedad Box-Sizing

La propiedad box-sizing es una propiedad CSS para cambiar el modelo de caja por defecto de los navegadores. Mas información: https://developer.mozilla.org/en-US/docs/CSS/box-sizing
El ancho de un elemento se altera si se le aplica un borde o un padding. Eso es porque la anchura del elemento que especificamos con CSS, por defecto no incluye borde ni padding.
Con box-sizing: border-box hacemos que el ancho especificado sea el equivalente al ancho total. Es decir, sumando borde y padding.
Las implicaciones son enormes, porque a la hora de realizar los cálculos para el Layout podemos incluir el tamaño que deseemos en el padding y border:

Veamos un ejemplo completo: **Ej16.06d-BoxSizing.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Centrado</title>
   <style type="text/css">
      #contenedor {     width: 600px;        height: 300px;
                        border: 3px solid blue;            }

      #contenedor div {       width: 200px;         height: 300px;
                        float: left;         padding: 5px;
                        box-sizing: border-box;
                        border: 2px solid red;             }

      #caja1 { background: pink;}
      #caja2 { background: aquamarine;}
      #caja3 { background: palegreen;}
    </style>
  </head>
  <body>
   <div id="contenedor">
      <div id="caja1">Caja1</div>
      <div id="caja2">Caja2</div>
      <div id="caja3">Caja3</div>
   </div>
  </body>
</html>
```

Si probamos a comentar **box-sizing: border-box;**
Veremos como la 3ª caja se descuadra, arruinando la estructura.

El valor **padding-box;**
Este valor no es tan útil como border-box, pero lo puede ser en ciertos casos. Lo que hace es que la anchura sea la del contenido y el padding (por lo que se le sumará el borde). Es experimental y sólo está implementada en Firefox.

16.6.5 Calc ()

Una alternativa a box-sizing es usar calc().
En este caso, creamos 3 cajas dentro de un contenedor y para su tamaño usamos un cálculo.
Mas información: http://ksesocss.blogspot.com/2012/03/funcion-calc-en-css-soporte-y-uso.html
https://developer.mozilla.org/es/docs/CSS/calc

Veamos un ejemplo modificando el anterior: **Ej16.06d-Calc.html**

```
<!DOCTYPE html>
...
      #contenedor div {
         width: -webkit-calc(600px/3 - 2px*2 - 5px*2);
         height: -webkit-calc(600px/3 - 2px*2 - 5px*2);
...
</html>
```

16.7 Otras propiedades

A fecha de hoy, 25 de Febrero de 2013, existen una serie de propiedades y avances de CSS3 que aún no han sido del todo implementados. Destacan especialmente:

A) Pestañas
Implementar pestañas con anchos personalizados.
MDN → https://developer.mozilla.org/es/docs/CSS/tab-size
Un par de ejemplos para crear pestañas (no con esta propiedad) en español:
http://vagabundia.blogspot.com/2010/04/tabs-solo-con-css.html
http://vagabundia.blogspot.com/2011/08/variante-de-pestanas-usando-solo-css.html

B) Regiones
Una idea excelente: conectar DIVs independientes indicándolo con una propiedad.
W3C → http://dev.w3.org/csswg/css3-regions/
Mas... → http://net.tutsplus.com/tutorials/html-css-techniques/diving-into-css-regions/

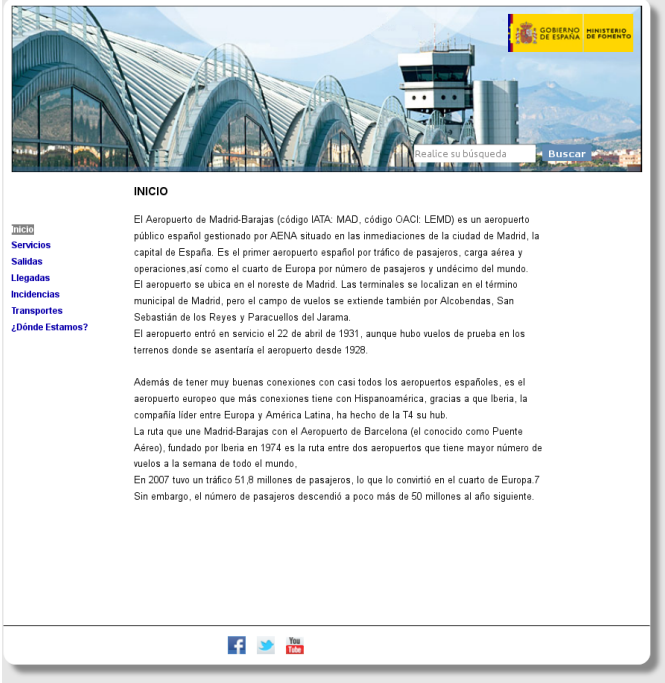
C) Plantillas (Estructura)
W3C → http://dev.w3.org/csswg/css3-layout/

D) Presentación en Rejilla
Ojo, solo Explorer 10!
http://www.w3.org/TR/css3-grid-layout/

17 PROYECTO WEB

Vamos a implementar la web del Aeropuerto Madrid Barajas que tendrá las siguientes páginas y características:

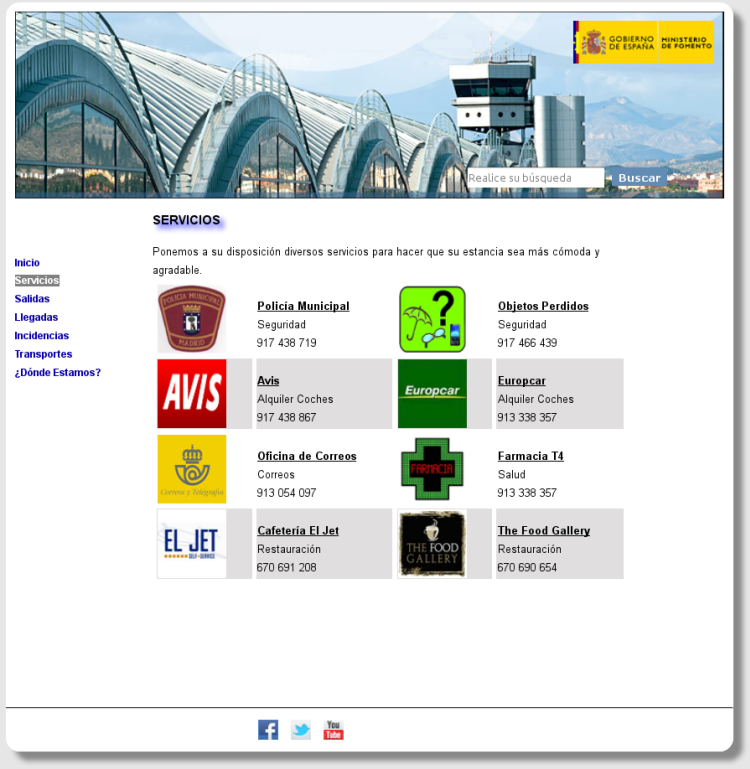
1. La página de inicio será:



Para el texto de presentación podemos usar el que viene en la Wikipedia.
Las imágenes las podemos tomar de Aena o bien mediante una búsqueda en Google.

A la izquierda vamos a incluir un menú vertical donde indicaremos donde nos encontramos cambiando la forma de la opción elegida (en este caso Inicio).

Incluiremos un buscador con el botón modificado situado en la esquina inferior derecha de la cabecera.



En dicha cabecera introduciremos un elemento independiente que será el logo del Ministerio de Fomento situado en la esquina superior derecha que nos llevará a la página de AENA.

2. La siguiente página será de Servicios.

Obsérvese como se ha añadido una sombra de texto al título SERVICIOS de color azul semitransparente.

Además, en la estructura general de la página, debemos incluir sendos botones hacía las páginas de AENA en Facebook, Twitter y Youtube, incluyendo sus respectivos iconos.

Es importante recordar algo esencial:
Las reglas CSS deben ir en un archivo aparte, debidamente comentadas y jerarquizadas, incluyendo, en la medida de lo posible, las rutas de los selectores al completo. Por ejemplo:
#cabecera #buscador #botonbusqueda

3. La página 3 será una tabla con las **SALIDAS**.

4. La página 4 serán las **LLEGADAS**; en ambos casos deben modificarse los colores para los estados CANCELADO y RETRASADO.

SALIDAS

Con el objetivo de ofrecer un servicio más ágil a los pasajeros, el aeropuerto de Madrid-Barajas ha llevado a cabo una redistribución de las compañías aéreas por terminales.

S/L	Código	Hora	Destino	Puerta	Estado
	IB 804	12:07 PM	(LPA) Las Palmas	01	EN HORA
	EW 3291	12:09 PM	(DUS) Dusseldorf	01	EN HORA
	AZ 61	12:11 PM	(FCO) Rome	01	EN HORA
	LH 1801	12:20 PM	(MUC) Munich	01	CANCELADO
	SU 3422	01:10 PM	(TFN) Tenerife	01	RETRASADO

Para mejorar la visualización de la página debemos hacer que las filas impares (sin contar la fila de títulos de la tabla) tengan un fondo ligeramente mas oscuro que las impares.

LLEGADAS

Con el objetivo de ofrecer un servicio más ágil a los pasajeros, el aeropuerto de Madrid-Barajas ha llevado a cabo una redistribución de las compañías aéreas por terminales.

S/L	Código	Hora	Origen	Puerta	Estado
	UX 9054	02:05 PM	(TFN) Tenerife	01	EN HORA
	IB 6126	01:35 PM	(BCN) Barcelona	02	EN HORA
	KL 3391	02:25 PM	(AMS) Amsterdam	01	EN HORA
	YW 8915	02:40 PM	(MRS) Marseille	02	CANCELADO
	YW 8711	02:06 PM	(LYS) Lyon	02	EN HORA
	AA 5716	02:32 PM	(ORY) Paris	04	EN HORA
	FR 5357	02:35 PM	(KRK) Krakow	01	EN HORA
	IB 6464	03:13 PM	(GYE) Guayaquil	05	RETRASADO

Evidentemente, tendremos que incluir los respectivos iconos de llegadas y salidas.

¡ATENCIÓN!
Estas páginas de Salidas, Llegadas y la siguiente de incidencias será empleadas mas adelante durante la parte del curso de PHP / MySQL.

Incluiremos, en una página de Administración aparte:

- Consulta
- Inserción
- Borrado
- Modificación

5. La página 5 será la página de Incidencias que, como el resto de elementos de tabla anteriores, tendrá colores diferentes en función de los estados, iconos específicos de Salidas y llegadas y fondos algo mas oscuros para elementos impares.

INCIDENCIAS

En este Panel se encuentran las Incidencias de Salidas/Llegadas del Aeropuerto Madrid - Barajas.

S/L	Código	Hora	Origen	Incidencia	Estado
	LH 1801	12:20 PM	(MUC) Munich	RETRASADO	SOLVENTADO
	SU 3422	01:10 PM	(TFN) Tenerife	CANCELADO	PENDIENTE
	YW 8915	02:40 PM	(MRS) Marseille	RETRASADO	SOLVENTADO
	IB 6464	03:13 PM	(GYE) Guayaquil	CANCELADO	PENDIENTE

TRANSPORTES

El Aeropuerto Madrid - Barajas tiene una amplia conexión de transportes con Madrid: Carretera (varias lineas de Bus), Metro y Cercanias RENFE.

Taxi
Paradas: T1 (PB y P1); T2 (PB y P2); T3 (PB); T4 (PB y P2)
Radio Teléfono Taxi: 915 478 200

Empresa Municipal de Transportes de Madrid
Paradas: T1, T2 y T4 (Planta Baja - PB)
Teléfono: 902 50 78 50

Metro de Madrid
Paradas: T4, planta -1. T2, primera planta
Atención al Cliente: 902 444 403

Cercanías Renfe
Paradas: T4, planta -1. Se puede llegar a la T123 a través de autobuses lanzaderas.
Atención al Cliente: 902 320 320

6. La página 6 incluirá los TRANSPORTES hasta la terminal. Se debe incluir los siguientes:
   - Taxi
   - EMT
   - Metro
   - Cercanias

   En todos los iconos incluiremos enlaces a sus respectivas páginas.

7. Por último, en la página 7, ¿DONDE ESTAMOS? Incluiremos un iframe de Google Maps al Aeropuerto de Barajas y debajo su dirección.

18 BIBLIOGRAFÍA Y SOLUCIONES

18.1 Bibliografía Recomendada

– El gran libro de HTML5, CSS3 y Javascript
  Juan Diego Gauchat
  Marcombo, 2012
  ISBN: 8426717829

– XHTML y CSS: Los nuevos estándares del código fuente [2ª edición]
  Luc VAN LANCKER
  Ediciones ENI
  ISBN: 978-2-7460-4742-6

18.2 Enlaces

– http://www.blooberry.com/indexdot/css/propindex/all.htm
  [ENG] Todas las propiedades CSS2 explicadas por orden.
– http://www.puntogeek.com/2013/02/14/como-convertir-tu-letra-escrita-a-mano-en-una-fuente-para-la-computadora/
  Crear fuentes a partir de nuestra letra a mano
– https://developer.mozilla.org/en-US/docs/CSS
  Página Oficinal de Mozilla Developer Network
–

18.3 Soluciones Ejercicios

Ejercicio07.html

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
   <meta charset="UTF-8" />
   <title>Ejercicio 4</title>
   <link href="css/Ejercicio07.css" rel="stylesheet" />
</head>
<body>
<section>
<!--Inicio Contenedor -->
<div id="contenedor">

      <!--inicio cabecera-->
      <div id="cabecera">
            <div id="logo">
               <h1> <span> </span></h1>
            </div>
            <div id="buscador">
                  <form action="php/Ejercicio04.php" method="post">
                        Buscar: <input type="search" placeholder="realice su busqueda" />
                  </form>
            </div>
            <div class="clear"></div>
      </div>
      <!--fin cabecera-->

      <!--inicio menu principal-->
      <div id="menu">
            <ul>
                  <li><a href="#"> enviar historia</a> |</li>
                  <li><a href="#"> portada</a> |</li>
                  <li><a href="#"> pendientes</a> |</li>
                  <li><a href="#"> populares</a> |</li>
                  <li><a href="#"> más visitadas</a> |</li>
                  <li><a href="#"> destacadas</a> |</li>
            </ul>
            <div class="clear"></div>
      </div>
      <!--fin menu principal-->

      <!--Inicio Lateral Izquierdo -->
      <div id="lateral">

            <!--Inicio Sección Noticias. Enlaces -->
            <div id="noticias">
                  <h3><a href="#">Noticias</a></h3>
                  <p><span class="fecha">dd/mm/aaaa</span></p>
                  <p><a href="#">Lorem ipsum dolor</a></p>
                  <p><span class="fecha">dd/mm/aaaa</span></p>
                  <p><a href="#">Lorem ipsum dolor</a></p>
                  <p><span class="fecha">dd/mm/aaaa</span></p>
                  <p><a href="#">Lorem ipsum dolor</a></p>
                  <p><span class="fecha">dd/mm/aaaa</span></p>
                  <p><a href="#">Lorem ipsum dolor</a></p>
                  <p><span class="fecha">dd/mm/aaaa</span></p>
                  <p><a href="#">Lorem ipsum dolor</a></p>

                  <br />
                  <h3><a href="#">Enlaces relacionados</a></h3>
                  <ul>
                        <li><a href="#"> ipsum</a></li>
                        <li><a href="#"> dolor</a></li>
                        <li><a href="#"> sit</a></li>
                  </ul>
            </div>
            <!--Fin Sección Noticias. Enlaces -->
```

```
      <!-- Inicio Sección Publicidad-->
      <div id="publicidad">
            <h3><a href="#">Publicidad</a></h3>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
            <p>Vivamus quis arcu massa.</p>
            <p>Etiam nec risus sit amet lorem tempor hendrerit.</p>
            <p><a href="#">Seguir leyendo...</a></p>
      </div>
      <!-- Fin Sección Publicidad-->

</div>
<!-- Fin Lateral Izquierdo-->

<!-- Inicio Contenido-->
      <div id="contenido">
      <!-- Inicio Cuerpo Principal-->
      <div id="principal">

            <!-- Inicio Artículo 1 -->
            <div class="articulo">
            <h3>El precio de la libertad de Android </h3>
            <img src="img/Android.jpg" alt="Android" />
            <p>Que Android es un sistema operativo libre (en el sentido de Open
            Source) es algo que estamos hartos de oir. Sin embargo a veces hay
            ciertas noticias que nos hacen dudar que esta libertad sea tal como en un
            principio puede parecer. Puesto que la principal diferencia de base de
            Android con respecto a sus competidores es ser 'libre', creemos que este
            tema merece una buena explicación para aclarar todo lo que hay detrás de
            esta supuesta libertad. </p>
                  <p>
            <a href="http://www.elandroidelibre.com/2013/01/el-precio-de-la-libertad-
            de-android.html"> Seguir leyendo...</a></p>
            </div>
            <!-- Fin Artículo 1 -->

            <!-- Inicio Artículo 2 -->
            <div class="articulo">
            <h3>Richard Stallman:
            "Tener el control de tu informática es un derecho humano" </h3>
            <img src="img/Stallman.jpg" alt="Dios Todopoderoso" />
            <p>"Ecuador es el mejor ejemplo de una política estatal de migración a
            software libre de las agencias públicas. Con el software libre los
            usuarios tienen el control del programa, si éste no es libre, es el
            programa quien controla a los usuarios. Cuando el usuario es el Estado,
            estamos hablando del control de su soberanía informática. Cualquier país
            debe migrar al software libre para echar al software privativo, ahora
            mismo Ecuador tiene la mejor política en ese sentido." </p>
            <p>
            <a href="http://www.lamarea.com/2013/01/30/tener-el-control-de-tu-
            informatica-es-un-derecho-humano/"> Seguir leyendo...</a></p>
            </div>
            <!-- Fin Artículo 2 -->
      </div>
      <!-- Fin Cuerpo Principal-->

      <!-- Inicio Cuerpo Secundario-->
      <div id="secundario">
            <h3>Las mejores catástrofes culinarias de Pinterest</h3>
            <p>La vida no es como aparece en Pinterest. Y la cocina, menos.
            Y la repostería, menos todavía. </p>
            <p><a href="#">Seguir leyendo...</a></p>
            <br />
            <h3>La madre que usó una tarjeta perdida para comprar pañales,
            a prisión en 15 días</h3>
            <p>Emilia Soria, la joven que hace seis años utilizó una tarjeta
            que se encontró en la calle para comprar pañales y comida para sus hijas,
            irá a la cárcel en un plazo máximo de quince días. </p>
            <p><a href="#">Seguir leyendo...</a></p>
      </div>
      <!-- Fin Cuerpo Secundario-->
```

```
      </div>
      <!-- Fin Contenido -->

      <div class="clear"></div>

      <!--Inicio Pie -->
      <div id="pie">
            <span class="enlaces">
                  <a href="#"> todas</a> |
                  <a href="#"> actualidad</a> |
                  <a href="#"> cultura</a> |
                  <a href="#"> ocio</a> |
                  <a href="#"> tecnología</a>
            </span>
            <span class="copyright">
                  &copy; 2013 Leonor Garzón
            </span>
            <div class="clear"></div>

      </div>
      <!--Fin Pie -->

</div>
<!--Fin Contenedor -->
</section>
</body>
</html>
```

Ejercicio07.css

```
/* ------- CONTENEDOR ---------*/
#contenedor {  width: 95%;    max-width: 900px;
               margin:0 auto;

               /*Ejercicio 7-a       */
               color: black;
               font-family: Arial;
               font-size: 0.9em;
               line-height: 1.4em;           }


#noticias, #publicidad, #principal, #secundario {
         padding: .5em;                              }

#cabecera, #menu {
         clear: both;                        }

/* ---------  CABECERA ---------*/
#cabecera { padding: 1em;                    }

#logo{      float:left;                              }

#logo h1 {  background-image: url("../img/logo.gif");
            background-position: -10px -10px;
            background-repeat: no-repeat;
            text-indent: 200px;
            height: 35px;                    }

#buscador{  float: right;                    }


/* -------- MENU SUPERIOR -----------*/
#menu {     padding: 0.5em;
            /*margin: .5em 0em;*/
            border-bottom: 1px solid #004C99;
            background-image: url("../img/fondo_menu.gif");
             /* Ejercicio 7c */
            margin: 1em;                     }
```

```
 #menu ul, #menu li{
            display: inline;
            float:left;                             }

/* Ejercicio 7c */
#menu a {   color: white;
            font-size: 1.3em;                       }


/* -------- CAJA LATERAL ----------- */

#lateral {  float: left;
            width: 20%;
            /* Ejercicio 7d */
            font-size: 1em;                 }

#lateral #noticias {
            border: 3px solid #C5C5C5;
            background-color: #D8D8D8;      }

#lateral #noticias h3 a {
            /* Ejercicio 7d */
            color: #003366;                 }


#lateral #noticias .fecha {
            /* Ejercicio 7d */
            display: block;
            color: #999;                    }

#lateral #publicidad {
         margin-top: 1em;
         border: 3px dashed #CC6600;
         background-color: #FFF6CD;
         /* Ejercicio 7d */
         color: #555;                       }

#lateral #publicidad a {
         /* Ejercicio 7d */
         color: #CC6600;                    }


/* ############ CAJA CONTENIDO ############# */

#contenido {
         float: right;
         width: 78%;                                }

/* --------- CAJA PRINCIPAL --------*/
#principal {
         float: left;
         width: 68%;                                }

img{     border: none;                      }

.articulo img{
         width: 100px;
         float: left;
         margin: .5em;                      }

.articulo {
         margin-top: 1em;
         font-size: 15px;                   }

#principal .articulo h3 {
         /* Ejercicio 7b*/
         color: #CC6600;
         font-size: 1.6em;
         line-height: 1.2em;
         margin-bottom: 0.3em;              }
```

```
#principal .articulo a {
            /* Ejercicio 7d*/
            color: #CC6600;                 }


/* -------------- COLUMNA DERECHA -----------*/
#secundario {
            float: right; /*se ha cambiado de left a right*/
            width: 25%;
            border: 3px solid #CC6600;      }

#secundario h3{
            background-color: #DB905C;
            padding: .2em;
            color: white;                   }


/* ############ FIN CAJA CONTENIDO ############# */


/* ------------ CAJA PIE ------------*/
#pie{
            clear: both;
            padding: .5em 0em;
            margin-top: 1em;
            border-top: 3px solid #C5C5C5;
            border-bottom: 3px solid #C5C5C5;

            /* Ejercicio 7e*/
            background-color: #D8D8D8;
            margin: 1em;
            color: white;                   }

#pie .enlaces{
            float:left;                             }

#pie .copyright{
            float: right;

            /* Ejercicio 7e*/
            color: black;
            font-weight: bold;
            margin-right: 10px;             }


/* ######## REGLAS GENÉRICAS ####### */

ul, ul li{  margin: 0;
            padding: 0;
            list-style: none;                       }


h1, h3, p, form{
            margin: 0;
            padding: 0;                             }

.clear {       clear: both;                 }
```

Ejercicio11.html

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
   <title>Ejercicio11</title>
<style type="text/css">
table {           font: .9em Arial, Helvetica, sans-serif;
                  border: 1px solid #333;
                  border-collapse: collapse;
                  text-align: center;        }

table th {        background: #F5F5F5 url(imagenes/fondo_gris.gif) repeat-x;
                  padding: 0 .3em;           }

table th.euro {   background: #E6F3FF url(imagenes/euro.png) no-repeat left center;
                  padding: 0 .3em 0 1.2em;   }

table th.dolar {  background: #E6F3FF url(imagenes/dolar.png) no-repeat left center;
                  padding: 0 .3em 0 1.2em;   }

table th.libra {  background: #E6F3FF url(imagenes/libra.png) no-repeat left center;
                  padding: 0 .3em 0 1.2em;   }

table th.yen {    background: #E6F3FF url(imagenes/yen.png) no-repeat left center;
                  padding: 0 .3em 0 1.2em;   }

table th, table, td {   border: 1px solid #333;
                        line-height: 2em;    }

.fila {  text-align: left;    }
.par  {  background-color:#FFFFCC;   }

/* Cambiamos colores al poner el cursor encima o pulsar */
table tr:hover {   background: aquamarine !important;      }
table tr:active {  background: cyan !important;            }
</style>
</head>

<body>
<table summary="Tipos de cambio">
  <tr>
    <th scope="col">Cambio</th>
    <th scope="col">Compra</th>
    <th scope="col">Venta</th>
    <th scope="col">Máximo</th>
    <th scope="col">Mínimo</th>
  </tr>
  <tr class="par">
    <th scope="row" class="fila euro">Euro/Dolar</th>
    <td>1.2524</td>    <td>1.2527</td>    <td>1.2539</td>    <td>1.2488</td>
  </tr>
  <tr>
    <th scope="row" class="fila dolar">Dolar/Yen</th>
    <td>119.01</td>    <td>119.05</td>    <td>119.82</td>    <td>119.82</td>
  </tr>
  <tr class="par">
    <th scope="row" class="fila libra">Libra/Dolar</th>
    <td>1.8606</td>    <td>1.8611</td>    <td>1.8651</td>    <td>1.8522</td>
  </tr>
  <tr>
    <th scope="row" class="fila yen">Yen/Euro</th>
    <td>0.6711</td>    <td>0.6705</td>    <td>0.6676</td>    <td>0.6713</td>
  </tr>
</table>
</body>
</html>
```

Ejercicio12.html

```
<!DOCTYPE html>
<html lang="es">
  <head> <meta charset="UTF-8" />
<title>Ejercicio12</title>
<style type="text/css">
/* Formatear el formulario a dos columnas */
body {   font: 13px/1.6 Tahoma, sans-serif;
         background: #F5F5F5;                }

.izquierda {   float: left;   clear: left;   }
.derecha {     float: right;  clear: right;  }

ul {     list-style: none;
         margin: 0;  padding: 0;      }

#contenedor {  background: #FFF;
               border: 1px solid silver;
               margin: 1em auto;
               padding: 1em;  width: 768px;  }

span.requerido {  font-size: 1.3em;
                  font-weight: bold; color: #C00;   }

h2 {     font: normal 2em arial, sans-serif;
         margin: 0;  }

ul li.botones {   border-top: 2px solid #CCC;
                  clear: both;       float: none;
                  padding: 1em 0;    margin-top: 1em;
                  text-align: right; width: 100%;   }

ul li.botones input {   font-size: 1.3em;    }

ul li {  margin: 0.5em 0;
         padding: 0.5em;      width: 46%;    }

ul li label.titulo {          font-weight: bold;    }
ul li div span {  float: left;       padding: 0.3em 0;     }

ul li div span.completo {     width: 100%;   }
ul li div span.mitad {        width: 50%;    }
ul li div span.tercio {              width: 33%;    }
ul li div span.dostercios {   width: 66%;    }

ul li div span label {        display: block;       color: #333;
                              font: normal 0.8em arial, sans-serif;       }

ul li p.ayuda {   display: none;     }
ul li input {     padding: 0.2em;    }

input#nombre, input#apellido1, input#apellido2,
input#direccion, input#email {              width: 260px;  }

input#codigopostal {    width: 80px;  }
select#provincia {   width: 90px;    }
input#municipio, select#pais {              width: 150px;  }
input#telefonofijo, input#telefonomovil {   width: 135px;  }

/* Cambiar el color en el :hover y resaltar los campos en el :focus Ejercicio2*/
ul li:hover {           background-color: #FF9;     }

ul li.botones:hover {   background-color: transparent;     }
ul li input:focus {     border: 2px solid #E6B700;         }
```

```
/* Formatear el formulario a una columna Ejercicio3*/
ul li.izquierda, ul li.derecha {
         float: none;  width: auto;  }

ul li {  overflow: hidden;    }

ul li label.titulo {          float: left;          width: 150px;  }
ul li div {                   margin-left: 160px;   }

/* Aspecto final del formulario con los mensajes de ayuda Ejercicio4*/
h2 {     margin-bottom: 0.3em;       }

ul li {  border-top: 1px solid #CCC;
         margin: 0;     padding: 1em;               }

ul li.botones {      margin: 0;              }
ul li label.titulo {    text-align: right;   width: 100px;  }

ul li div {       margin-left: 110px;        overflow: hidden;     }
ul li {        position: relative;   }

ul li:hover p.ayuda {
         display: block;      margin: 0.3em;
         position: absolute;  width: 150px;
         top: 0;              right: 0;
         color: red;                 font-weight: bold;    }
</style>
</head>

<body>
<div id="contenedor">
<h2>Formulario de alta</h2>
<form method="post" action="#">
<ul>
<li class="izquierda">
  <label class="titulo" for="nombre">Nombre y apellidos <span
class="requerido">*</span></label>
  <div>
    <span class="completo">
      <input id="nombre" name="nombre" value="" />
      <label for="nombre">Nombre</label>
    </span>

    <span class="completo">
      <input id="apellido1" name="apellido1" value="" />
      <label for="apellido1">Primer apellido</label>
    </span>

    <span class="completo">
      <input id="apellido2" name="apellido2" value="" />
      <label for="apellido2">Segundo apellido</label>
    </span>
  </div>

  <p class="ayuda">No te olvides de escribir también tu segundo apellido</p>
</li>

<li class="derecha">
  <label class="titulo" for="direccion">Dirección <span class="requerido">*</span></label>

  <div>
    <span class="completo">
      <input id="direccion" name="direccion" value="" />
      <label for="direccion">Calle, número, piso, puerta</label>
    </span>

    <span class="tercio">
      <input id="codigopostal" name="codigopostal" value="" />
      <label for="codigopostal">Código postal</label>
    </span>
```

```
    <span class="dostercios">
      <input id="municipio" name="municipio" value="" />
      <label for="municipio">Municipio</label>
    </span>

    <span class="tercio">
      <select id="provincia" name="provincia">
        <option value=""></option>
        <option value="provincia1">Provincia 1</option>
        <option value="provincia2">Provincia 2</option>
        <option value="provincia3">Provincia 3</option>
      </select>
      <label for="provincia">Provincia</label>
    </span>

    <span class="dostercios">
      <select id="pais" name="pais">
        <option value=""></option>
        <option value="pais1">País 1</option>
        <option value="pais2">País 2</option>
        <option value="pais3">País 3</option>
      </select>
      <label for="pais">País</label>
    </span>
  </div>

  <p class="ayuda">El código postal es imprescindible para poder recibir los pedidos</p>
</li>

<li class="izquierda">
  <label class="titulo" for="email">Email</label>

  <div>
    <span class="completo">
      <input id="email" name="email" value="" />
    </span>
  </div>

  <p class="ayuda">Asegúrate de que sea válido</p>
</li>

<li class="derecha">
  <label class="titulo" for="telefonofijo">Teléfono <span
class="requerido">*</span></label>

  <div>
    <span class="mitad">
      <input id="telefonofijo" name="telefonofijo" value="" />
      <label for="telefonofijo">Fijo</label>
    </span>

    <span class="mitad">
      <input id="telefonomovil" name="telefonomovil" value="" />
      <label for="telefonomovil">Móvil</label>
    </span>
  </div>

  <p class="ayuda">Sin prefijo de país y sin espacios en blanco</p>
</li>

<li class="botones">
  <input id="alta" type="submit" value="Darme de alta &rarr;" />
</li>

</ul>
</form>

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

Ejercicio13.html

```
<!DOCTYPE html> <html lang="es">
<head> <meta charset="UTF-8" />
   <title>Ejercicio13</title>
   <link rel="stylesheet" type="text/css" href="Ejercicio13.css" />
</head>
<body>
   <div id="contenedor">
      <div id="cabecera">
         <div id="logo">
            <h1><a href="#">Mi sitio web</a></h1>
         </div>
         <div id="buscador">
            <form action="" method="post" name="buscar">
               <input id="campobusqueda" type="text" name="busqueda"
                     placeholder="Realice su búsqueda"/>
               <input type="submit" name="" value="Buscar"/>
            </form>
         </div>
      </div>

      <div id="menu">
         <ul>
            <li><a href="#">Inicio</a></li>          <li><a href="#">Blog</a></li>
            <li><a href="#">Sobre mí</a></li>        <li><a href="#">Fotos</a></li>
            <li><a href="#">Portfolio</a></li>       <li><a href="#">Contacto</a></li>
            <li><a href="#">Enlaces</a></li>
         </ul>
      </div>

      <div id="contenido">

         <div id="principal">

            <div class="articulo">
         <h2><a href="#">Diez palabras que no sabías que eran marcas registradas</a></h2>
               <p class="info">
                  <span class="fecha">dd-mm-aaaa hh:mm</span>
                  <span class="categoria"><a href="#">categoria</a></span>
                  <span class="autor"><a href="#">autor</a></span>
                  <span class="comentario"><a href="#">comentario</a></span>
               </p>
<p>El mundo de la propiedad intelectual causa dolores de cabeza a creadores y abogados del
mundo entero. Cada territorio cuenta con sus propias leyes, y lo que en EEUU puede ser
aceptado como "marca registrada", aquí puede encontrarse con la negativa de los organismo
pertinentes. Aún así, todas las palabras o expresiones que os presentamos son propiedad de
alguna empresa que vigila de manera cuidadosa su utilización. En algunos casos, la
ubicuidad de algunas marcas comerciales las terminan por hacerlas de uso común:</p>
               <p class="leermas"><a href="#">Seguir leyendo...</a></p>
            </div>

            <div class="articulo">
               <h2><a href="#">Disponible LibreOffice 4.0 v. final para descargar</a></h2>
               <p class="info">
                  <span class="fecha">dd-mm-aaaa hh:mm</span>
                  <span class="categoria"><a href="#">categoria</a></span>
                  <span class="autor"><a href="#">autor</a></span>
                  <span class="comentario"><a href="#">comentario</a></span>
               </p>
<p>The Document Foundation acaba de anunciar la disponibilidad de LibreOffice 4.0,
la última versión de la suite ofimática libre. Entre las novedades de esta versión,
entre otras muchas se incluye mejor interoperabilidad con formatos docx, rtf, visio
y publisher, soporte de "temas" de interfaz (las "personas" de Mozilla),soporte para
el estándar CMIS para sistemas de gestión documental y una revisión completa del
código fuente que servirá de base para su desarrollo futuro. </p>
               <p class="leermas"><a href="#">Seguir leyendo...</a></p>
            </div>
         </div><!--Fin Div Principal-->
```

```
         <div id="secundario">
            <h3><a href="#">Sobre mí</a></h3>
            <div id="sobremi">
               <img src="imagenes/ivan-rodriguez-ruiz.jpg" alt="mi foto"/>
               <p><strong>Iván Rodríguez</strong><br />
               Edad: 36<br />
               Ciudad: Sevilla<br />
               <a href="#">Mi perfil público</a>
               </p>
            </div>

            <h3>Categorías</h3>
            <ul>
               <li><a href="#">Categoría 1</a></li>        <li><a href="#">Categoría
2</a></li>
               <li><a href="#">Categoría 3</a></li>        <li><a href="#">Categoría
4</a></li>
               <li><a href="#">Categoría 5</a></li>        <li><a href="#">Categoría
6</a></li>
            </ul>

            <h3>Archivo</h3>
            <ul>
               <li><a href="#">Febrero 2013</a></li>
               <li><a href="#">Enero 2013</a></li>
               <li><a href="#">Diciembre 2012</a></li>
               <li><a href="#">Noviembre 2012</a></li>
               <li><a href="#">Octubre 2012</a></li>
            </ul>

            <h3> </h3>
            <img src="imagenes/pie_secundario.jpg" alt="pie Secundario" />
         </div><!--Fin Div Secundario-->

      </div><!--Fin Div Contenido-->

      <div id="pie">
         <p id="copyright">&copy; 2013 Iván Rodríguez</p>
      </div><!--Fin Div Pie-->

   </div><!--Fin Div Contenedor-->
</body>
</html>
```

Ejercicio13.css

```
/* Estos son las Reglas CSS del Ejercicio 13.html */
/* Reglas generales de la página */
body {         margin: 0;            padding: 0;
               font-size: 70%;       font-family: verdana;
               background: url("imagenes/fondo.gif") 0 0 repeat-x;        }

a {               color: #192666;     }
a:hover {      color: #4f6ad7;       }

p {               margin: 1em 0;             padding: 0;    }
h1,h2,h3 {     margin: 1em 0;                padding: 0;    }

h1 {           font-size: 260%;      font-weight: normal;
               font-family: georgia;                }

h2 {           font-size: 180%;      font-weight: normal;   }
h3 {           font-size: 120%;      }


/* ---------------------------- Estructura o Layout ----------------------------  */

#contenedor {  width: 770px;         margin: 0 auto;       }
#cabecera {       width: 770px;             height: 100px;
               margin-top: 20px;     padding: 0;
```

```
                background: #233c9b
                url("imagenes/cabecera.jpg") no-repeat;      }

 #menu {          margin: 0 5px;            padding-top: 10px;
                overflow: hidden;
                background-color: #192666;                  }

#contenido {   width: 760px;         margin: 0 5px;               }
#contenido #principal {
                float: left;          width: 530px;
                margin-top: 15px;     padding-left: 20px;
                background: white;                          }

#contenido #secundario {
                float: left;          width: 200px;
                margin-top: 15px;     padding-left: 0;
                background: #cedbf9
                url("imagenes/fondo_columna.gif") no-repeat;        }

/* ---------------------------- Cabecera ----------------------------  */

#cabecera #logo {
                margin-left: 50px;    float: left;          }

#cabecera #logo a {
                color: white;         line-height: 35px;    }

#cabecera #logo a:hover {
                color: #b5c4e3;       text-decoration: none; }

#cabecera #buscador {
                float: right;         margin: 30px;         }

#cabecera #buscador #campobusqueda {        width: 250px;         }

/* ---------------------------- Menú ----------------------------  */

#menu ul {      margin: 0 10px;       padding: 0;
                list-style: none;                           }

#menu ul li {   margin-right: 5px;
                padding: 0;           float: left;          }

#menu ul li a{    display: block;     position: relative;
                color: #b5c4e3;       padding: 5px 5px;
                background-color: #253575;
                font-weight: bold;
                text-decoration: none;
}

#menu ul li a:hover {                 background-color: white;
                color: orange;                                      }

/* ---------------------------- Contenido ----------------------------  */

#contenido .articulo {                margin: 20px 20px 0 0;
                background: url("imagenes/fondo_articulo.jpg") no-repeat;
                padding: 20px;                                      }

#contenido .articulo h2 {             margin: 0 -20px;
                padding: 10px;                background: #dee5fd;
                color: #192666;                             }

#contenido .info {                    margin: 10px 0;
                color: #6685cc;       padding-bottom: 8px;
                border-bottom: 1px solid #dee5fd;           }

#contenido .info a:hover {            color: #ff9000;       }
#contenido .info span {                       margin-left: 10px;    }
```

```
#contenido .info span.fecha, #contenido .info span.categoria,
#contenido .info span.autor, #contenido .info span.comentario {
               padding-left: 15px;                          }
 #contenido .info span.fecha {
               background: url("imagenes/icono_fecha.gif")
               0 50% no-repeat;                                     }

#contenido .info span.categoria {
               background: url("imagenes/icono_categoria.gif")
               0 50% no-repeat;                                     }

#contenido .info span.autor {
               background: url("imagenes/icono_autor.gif")
               0 50% no-repeat;                                     }

#contenido .info span.comentario {
               background: url("imagenes/icono_comentarios.gif")
               0 50% no-repeat;                                     }

/*------------secundario------------*/

#contenido #secundario h3 {
               margin-top: 20px;      padding: 10px 0 10px 10px;
               color: #192666;       background-color: #A0B9F3;    }

#contenido #secundario #sobremi {    padding: 0 10px;              }
#contenido #secundario #sobremi img {
               float: left;          margin: 4px 5px 0 0;          }

#contenido #secundario ul {          list-style: none;
               margin: 15px 0;       padding: 0;                   }

#contenido #secundario ul li {       border-bottom: 1px solid #E0E8FA;
               margin: 0;            padding: 0;                   }

#contenido #secundario ul li:hover {        background-color: #E0E8FA;    }

#contenido #secundario ul li a {
               display: block;       padding: 3px 0 3px 20px;
               text-decoration: none;
               background: url("imagenes/icono_elemento.gif")
               8px no-repeat;                                            }

#contenido #secundario ul li a:hover {
               color: #192666;
               background: url("imagenes/icono_elemento_seleccionado.gif")
               8px no-repeat;                                       }


/*---------------pie-------------*/

#pie {
   clear:both;
}
```