CSS

PID_00253480

Ben Buchanan
Christian Heilmann
Ben Henick
Tommy Olsson
Nicole Sullivan

Tiempo mínimo previsto de lectura y comprensión: 18 horas

Universitat Oberta de Catalunya

Índice

1. Conceptos básicos de CSS

Christian Heilmann

En esta asignatura ya hemos hablado sobre el contenido de los sitios web y cómo estructurar el contenido con HTML. Esto es muy importante, ya que quiere decir que damos significado y estructura a nuestros documentos para que otras tecnologías se puedan relacionar con ellos sin problemas. La tecnología web más importante que discutiremos a continuación es el CSS (*Cascading Style Sheets*, hojas de estilos en cascada), que se utiliza para aplicar estilos al HTML y situarlo en la página web. En este apartado, hablaremos del CSS: qué es, cómo aplicarlo al HTML y cuál es su sintaxis básica.

Podéis ver cómo estructurar el contenido de los sitios web en el módulo "Fundamentos de HTML".

1.1. ¿Qué es el CSS?

Mientras que el HTML estructura el documento e indica a los navegadores cuál es la función de un elemento concreto (¿es un vínculo hacia otra página?, ¿es un título?), el CSS da instrucciones al navegador sobre cómo debe mostrar un elemento concreto: estilo, espaciado y posición. Si el HTML son los puntales y los ladrillos que forman la estructura de una casa, el CSS es el yeso y la pintura que la decoran.

Esto se consigue al utilizar un sistema de reglas cuya sintaxis exacta veremos más abajo. Estas reglas dicen qué elementos de HTML deben tener estilos añadidos, y en cada regla enumeran las propiedades (por ejemplo, color, tamaño, tipo de letra, etc.) de aquellos elementos HTML que quieren manipular y los valores nuevos que les quieren aplicar.

Ejemplo

Una regla del CSS podría decir, por ejemplo: "Quiero encontrar todos los elementos `h2` y aplicarles el color verde", o: "Quiero encontrar todos los párrafos con el nombre de clase `author-name`, aplicarles un fondo de color rojo, hacer que el texto de su interior sea el doble de grande que el texto de los párrafos normales y añadir 10 píxeles de espaciado a su alrededor".

El CSS no es un lenguaje de programación como JavaScript ni tampoco es un lenguaje de etiquetas como HTML; de hecho, no hay nada con lo que se le pueda comparar. Las tecnologías que definían las interfaces antes del desarrollo de la web mezclaban siempre la presentación y la estructura. Sin embargo, en un entorno que cambia tan a menudo como la web, ésta no es una manera muy inteligente de hacer las cosas, y por ello se inventó el CSS.

1.2. Definir las reglas de los estilos

Así pues, veamos un ejemplo de código del CSS y después lo analizaremos:

```
selector {
  propiedad1:valor;
  propiedad2:valor;
  propiedad3:valor;
}
```

Las partes pertinentes son las siguientes:

- El selector identifica los elementos del HTML a los que se aplicará la regla, que se identifican con el nombre del elemento en sí, como por ejemplo `body`, o con algún otro método, como los valores del atributo `class`; ya volveremos a hablar de ello más adelante.

- Las llaves contienen las parejas de propiedad/valor, que se separan entre ellas con un signo de punto y coma; las propiedades se separan de sus valores respectivos con el signo de dos puntos.

- Las propiedades definen lo que queréis hacer con los elementos que habéis seleccionado. Pueden ser muy variadas y pueden modificar el color, el color de fondo, la posición, los márgenes, la separación, el tipo de letra y muchos otros aspectos del elemento.

- Los valores son los valores a los que queréis cambiar cada una de las propiedades de los elementos seleccionados. Los valores dependen de la propiedad; por ejemplo, las propiedades que afectan al color pueden tener valores hexadecimales como #336699, valores RGB como rgb (12,134,22) o nombres de colores (en inglés) como *red*, *green* o *blue*. Las propiedades que afectan a la posición, los márgenes, la anchura, la altura... se pueden medir en píxeles, ems, porcentajes, centímetros u otras unidades similares.

Veamos ahora un ejemplo concreto:

```
p {
  margin:5px;
  font-family:arial;
  color:blue;
}
```

El elemento HTML que selecciona esta regla es `p`; esta regla se aplicará a todos los `p` de los documentos HTML que utilicen este CSS, a menos que haya reglas más concretas que se les apliquen, ya que en este caso las reglas más concretas tendrán prioridad sobre esta regla. Las propiedades que se ven afectadas por esta regla son los márgenes alrededor de los párrafos, el tipo de letra del texto de los párrafos y el color de este texto. Los márgenes están definidos en 5 píxeles, el tipo de letra está definido en Arial y el color del texto es azul.

Ya volveremos a hablar de todos estos detalles más adelante; el objetivo principal de este apartado es explicar las bases del CSS y no sus detalles menores.

Todas estas reglas se unen para formar una hoja de estilos. Ésta es la sintaxis más básica de CSS. Hay más cosas, pero no muchas más; probablemente, lo mejor que tiene CSS es su sencillez.

1.2.1. Comentarios en CSS

Una de las primeras cosas que hay que saber es cómo hacer comentarios en CSS. Podéis añadir comentarios poniéndolos entre `/*` y `*/`. Los comentarios pueden ocupar varias líneas, que el navegador ignorará:

```
/* Éstos son selectores de elementos básicos */
selector{
  propiedad1:valor;
  propiedad2:valor;
  propiedad3:valor;
}
```

Podéis añadir comentarios entre reglas o en un bloque de propiedades; por ejemplo, en el siguiente CSS las propiedades segunda y tercera se encuentran entre delimitadores de comentario, por lo que el navegador las ignorará. Esto puede ser muy útil cuando queráis comprobar los efectos que tienen partes concretas del CSS en vuestra página web; podéis eliminarlas convirtiéndolas en comentarios, guardar el CSS y volver a cargar el HTML.

```
selector{
  propiedad1:valor;
  /*
  propiedad2:valor;
  propiedad3:valor;
  */
}
```

A diferencia de otros lenguajes, el CSS sólo tiene comentarios de bloque; los comentarios de línea no existen. Evidentemente, si lo queréis, podéis restringir el comentario a una única línea, pero deberéis seguir incluyendo los delimitadores de apertura y de cierre de comentario (`/*` y `*/`).

1.2.2. Agrupar selectores

También podéis agrupar diferentes selectores. Pongamos por caso que queréis aplicar el mismo estilo a `h1` y `p`; podríais escribir el siguiente CSS:

```
h1 {color:red}
p {color:red}
```

Sin embargo, ésta no es la manera ideal de hacer las cosas, ya que repetís información idéntica. Por lo tanto, podéis acortar el CSS agrupando los selectores con una coma; las reglas entre llaves se aplican a ambos selectores:

```
h1, p {color:red}
```

Hay varios selectores, y cada uno se corresponde con una parte diferente del etiquetado. Los tres más básicos que encontraréis más a menudo son los siguientes:

- `p{}`: selector de elementos.
  Se corresponde con todos los elementos de este nombre de la página (en el caso anterior, elementos `p`).

- `.example{}`: selector de clase
  Se corresponde con todos los elementos que tienen el atributo `class` con el valor especificado; por lo tanto, el selector anterior se correspondería con `<p class="example">`, `<li class="example">` o `<div class="example">`, o con cualquier otro elemento con `class` definido como `example`. Tened en cuenta que los selectores de clase no comprueban ningún nombre de elemento concreto.

- `#example{}`: selector de `id`.
  Se corresponde con todos los elementos que tienen un atributo `id` con el valor especificado; así pues, el selector anterior se correspondería con `<p id="example">`, `<li id="example">` o `<div id="example">`, o con cualquier otro elemento con `id` definido como `example`. Tened en cuenta que los selectores de ID no comprueban ningún nombre de elemento y que sólo podéis tener un ID de cada por documento HTML, ya que son únicos para cada página.

En los siguientes ejemplos podéis ver los selectores anteriores en acción. Observad que cuando abrís el ejemplo en un navegador, el estilo `warning` se aplica tanto al punto de la lista como al párrafo (si desaparece el pico, es porque definís un color de texto blanco sobre un fondo blanco).

- "Ejemplo de selectores"*
- "Selectors.css"**

*** http://mosaic.uoc.edu/ac/le/operafiles/27/example-selectors.html**

**** http://mosaic.uoc.edu/ac/le/operafiles/27/selectors.css**

También podéis unir varios selectores para definir unas reglas aún más específicas:

- `p.warning{}`. Se corresponde con todos los párrafos cuya `class` es `warning`.

- `div#example{}`. Se corresponde con el elemento con `example` como atributo `id`, pero sólo cuando es un `div`.

- `p.info, li.highlight{}`. Se corresponde con los párrafos cuya `class` sea `info` y con los puntos de lista con el valor `highlight` en `class`.

En el siguiente ejemplo, utilizamos selectores específicos para diferenciar los diferentes estilos de advertencias.

- "example-specificselectors.html"*
- "specificselectors.css"**

*** http://mosaic.uoc.edu/ac/le/operafiles/27/example-specificselectors.html**

**** http://mosaic.uoc.edu/ac/le/operafiles/27/specificselectors.css**

1.3. Selectores avanzados de CSS

En el subapartado anterior hemos explicado los selectores más básicos de CSS, que son los selectores de elemento, clase e identificador. Con estos selectores se pueden hacer muchas cosas, pero eso no lo es todo; hay otros selectores que permiten seleccionar elementos para aplicarlos en función de criterios más específicos:

- Selectores universales: los selectores universales se pueden utilizar para seleccionar todos los elementos de la página.

- Selectores de atributo: tal como indica su nombre, los selectores de atributos permiten seleccionar elementos en función de sus atributos.

- Selectores de hijos: si queréis seleccionar elementos concretos que son hijos de otros elementos concretos, podéis utilizar este selector.

- Selectores de descendientes: si queréis seleccionar elementos concretos que son descendientes de otros elementos concretos (no sólo hijos directos, sino también más abajo del árbol), podéis utilizar este tipo de selector.

- Selectores de hermanos adyacentes: si queréis seleccionar sólo elementos concretos que siguen a otros elementos concretos, utilizad estos selectores.

- Pseudoclases: permiten aplicar estilos a elementos, no en función de qué son los elementos, sino en función de factores más raros como, por ejemplo, el estado de los enlaces (por ejemplo, si se está encima o si ya se han visitado).

- Pseudoelementos: permiten aplicar estilos a partes concretas de los elementos y no a todo el elemento (por ejemplo, la primera letra de este elemento); también os permiten insertar contenido antes o después de elementos concretos.

Nota

A medida que vayáis progresando por esta asignatura, encontraréis referencias a algunos de estos selectores más complicados. Si no los entendéis a la primera, no debéis preocuparos; ya los iréis dominando a medida que ganéis experiencia en los estilos de las páginas web. Lo mejor es empezar con los tres selectores básicos mencionados en el subapartado anterior e ir utilizando los otros una vez que os vayáis sintiendo más seguros.

1.3.1. Selectores universales

Para decirlo rápidamente, los selectores universales seleccionan todos los elementos de una página para aplicarles estilos. Por ejemplo, la regla siguiente dice que todos los elementos de la página deben tener un borde sólido de color negro de 1 píxel:

```
* {
  border: 1px solid #000000;
}
```

1.3.2. Selectores de atributos

Los selectores de atributos permiten seleccionar elementos en función de los atributos que contienen. Por ejemplo, con el selector siguiente podéis seleccionar todos los elementos `img` con un atributo `alt`:

```
img[alt]{
  border: 1px solid #000000;
}
```

Tened en cuenta los corchetes.

Con el selector anterior quizá podríais definir un borde negro en torno a todas las imágenes con un atributo `alt` y un borde rojo alrededor de todas las demás imágenes, algo muy útil para las pruebas de accesibilidad.

Pero los atributos os pueden ser mucho más útiles si tenéis en cuenta que podéis hacer selecciones a partir del valor de los atributos, y no sólo a partir de sus nombres. La regla siguiente se aplica a todas las imágenes que tienen un atributo `src` con el valor `alert.gif`:

```
img[src="alert.gif"]{
  border: 1px solid #000000;
}
```

Quizá penséis que no es muy útil, pero puede serlo mucho para la depuración. Y aún mucho más útil es la posibilidad de seleccionar partes concretas de atributos, como por ejemplo extensiones de archivos. Y eso ya está llegando: CSS 3 introduce tres tipos nuevos de selectores de atributos que pueden hacer selecciones en función de las cadenas de texto de los valores de los atributos (al principio, al final o en cualquier punto del valor). Podéis leer el artículo de Christopher Schmitt sobre los selectores de atributos del CSS 3*.

*** http://dev.opera.com/articles/view/css-3-attribute-selectors/**

1.3.3. Selectores de hijos

Si queréis seleccionar elementos concretos que son hijos de otros elementos concretos, podéis utilizar este selector. Por ejemplo, la regla siguiente pondrá de color azul el texto de los elementos `strong` hijos de elementos `h3`, pero ninguno de los demás elementos `strong`:

```
h3 > strong {
  color: blue;
}
```

Nota

IE 6 y sus versiones anteriores no aceptan los selectores de hijos.

1.3.4. Selectores de descendientes

Los selectores de descendientes son muy similares a los selectores de hijos, excepto que estos últimos seleccionan sólo a los descendientes directos; los selectores de descendientes seleccionan los elementos pertinentes en cualquier punto de la jerarquía del elemento, y no sólo los descendientes directos. Fijémonos bien en qué quiere decir esto. Pongamos por caso el siguiente fragmento de HTML:

```
<div>
  <em>hello</em>
  <p>In this paragraph I will say
    <em>goodbye</em>.
  </p>
</div>
```

En este fragmento, el elemento `div` es el padre de todos los demás. Tiene dos hijos, un `em` y un `p`. El elemento `p` tiene un único elemento hijo, que es otro `em`.

Podéis utilizar un selector de hijos para seleccionar sólo el elemento `em` que hay inmediatamente dentro de `div`, de la siguiente manera:

```
div > em {
 ...
}
```

Si utilizáis un selector de descendientes de la siguiente manera:

```
div em {
 ...
}
```

entonces se seleccionarán los dos elementos `em`.

1.3.5. Selectores de hermanos adyacentes

Estos selectores permiten seleccionar un elemento concreto que aparece directamente después de otro elemento concreto en el mismo nivel de la jerarquía del elemento. Por ejemplo, si quisierais seleccionar todos los elementos `p` que aparecen inmediatamente después de los elementos `h2`, pero no los demás elementos `p`, podríais utilizar la siguiente regla:

```
h2 + p {
 ...
}
```

> **Nota**
>
> IE 6 y sus versiones anteriores no aceptan los selectores de hermanos adyacentes.

1.3.6. Pseudoclases

Las pseudoclases se utilizan para definir estilos, no para los elementos, sino para varios estados de los elementos. El uso más habitual que haréis de ellas es para aplicar estilos en los estados de los enlaces; por lo tanto, son los que veremos en primer lugar. La lista siguiente especifica las diferentes pseudoclases y describe el estado del enlace que seleccionan:

- `:link`: el estado normal por defecto de los enlaces, tal como se encuentran cuando se ven por primera vez.

- `:visited`: enlaces que ya habéis visitado en el navegador que estáis utilizando.
- `:focus`: enlaces (o campos de formularios, o cualquier otra cosa) que tienen en este momento el cursor del teclado en su interior.
- `:hover`: enlaces que tienen en este momento el puntero del ratón sobre ellos.
- `:active`: un enlace en el que se está haciendo clic.

Las siguientes reglas CSS definen que, por defecto, los enlaces sean de color azul (el valor por defecto en la mayoría de los navegadores). Al poneros encima, desaparece el subrayado del enlace. También queremos conseguir este mismo efecto cuando se selecciona el enlace con el teclado, por lo que duplicamos la regla `:hover` con `:focus`. Una vez que ya se ha visitado un enlace, éste queda de color gris. Por último, cuando un vínculo está activo, aparece en negrita para dar una pista adicional de que pasará algo.

```
a:link{
   color: blue;
}
a:visited{
   color: gray;
}
a:hover a:focus{
   text-decoration: none;
}
a:active{
   font-weight: bold;
}
```

Nota

Cuidado si no definís estas reglas en el orden en el que aparecen en el ejemplo anterior, ya que de otro modo puede ser que no funcionen de la manera esperada. Esto es por el modo como la especificidad hace que las reglas posteriores de la hoja de estilos anulen las reglas anteriores. En el siguiente apartado, aprenderemos más detalles sobre la especificidad.

La pseudoclase `:focus` también es útil como contribución a la usabilidad en los formularios. Por ejemplo, podéis destacar el campo de introducción de datos que tiene el cursor intermitente activo en su interior con una regla como la siguiente:

```
input:focus {
   border: 2px solid black;
background color: lightgray;
}
```

A continuación, explicaremos `:first-child`. Esta pseudoclase selecciona cualquier aparición del elemento que es el primer elemento hijo de su padre; de modo que, por ejemplo, la regla siguiente selecciona el primer punto (con pico o numerado) de cualquier lista y pone el texto en negrita:

```
li:first-child {
 font-weight: bold;
}
```

Finalmente, comentaremos brevemente la pseudoclase `:lang`, que selecciona elementos que tienen un idioma concreto definido con el atributo `lang`. Por ejemplo, el elemento siguiente:

```
<p lang="en-US">A paragraph of American text, gee whiz!<p>
```

Se podría seleccionar de la manera siguiente:

```
p:lang(en-US) {
 ...
}
```

1.3.7. Pseudoelementos

Los pseudoelementos tienen dos finalidades. En primer lugar, los podéis utilizar para seleccionar la primera letra o la primera línea del texto del interior de un elemento concreto. Para crear una letra capitular al principio de cada párrafo del documento, podéis utilizar la regla siguiente:

```
p:first-letter {
  font-weight: bold;
  font-size: 300%
  background-color: red;
}
```

La primera letra de cada párrafo aparecerá en negrita, un 300% más grande que el resto del párrafo y con un fondo rojo.

Para hacer que la primera línea de cada párrafo aparezca en negrita, podéis utilizar la regla siguiente:

```
p:first-line {
  font-weight: bold;
}
```

El segundo uso de los pseudoelementos es generar contenido por medio del CSS, lo cual es más complicado. Podéis utilizar los pseudoelementos `:before` o `:after` para especificar el contenido que se debe insertar antes o después del elemento que seleccionéis. Entonces podéis especificar qué es lo que queréis insertar. Como ejemplo muy sencillo, podéis utilizar la regla siguiente para insertar una imagen decorativa después de cada uno de los enlaces de la página:

```
a:after{
  content: " " url(flower.gif);
}
```

También podéis utilizar la función `attr()` para insertar los valores de los atributos de los elementos después del elemento. Por ejemplo, con la regla siguiente podéis insertar el destino de cada uno de los enlaces del documento entre paréntesis después del enlace:

```
a:after{
  content: "(" attr(href) ")";
}
```

Las reglas de este tipo son ideales para hojas de estilos de impresión, que son hojas de estilos que podéis escribir y que se aplican automáticamente cuando un usuario imprime una página. La ventaja para el usuario es que podéis ocultar toda la navegación que un usuario no podrá seguir en una copia impresa y utilizar la técnica anterior para que el lector pueda ver las URL a las que se hace referencia en una página.

También podéis insertar valores numéricos incrementados después de los elementos que se repiten (como por ejemplo picos o párrafos) con la función `counter()`; esto se explica con mucho más detalle en el artículo de dev.opera.com sobre los contadores de CSS*.

*** http://dev.opera.com/articles/view/automatic-numbering-with-css-counters/**

Nota

IE 6 y sus versiones anteriores no aceptan estos selectores. También debéis tener en cuenta que no debéis dar información importante con CSS, ya que el contenido no estará disponible para las tecnologías de asistencia o si un usuario decide no utilizar vuestra hoja de estilos. La norma básica es que el CSS es para los estilos y el HTML es para el contenido importante.

1.4. CSS abreviado

Otro detalle con el que os encontraréis normalmente en esta asignatura es el CSS abreviado. Es posible combinar varias propiedades de CSS relacionadas en una única propiedad para ahorraros tiempo y esfuerzo. En este apartado, veremos los tipos de propiedades abreviadas disponibles.

Nota

En este subapartado ya hemos utilizado el CSS abreviado sin decirlo. La regla `border: 2px solid black;` es la regla abreviada para especificar por separado `border-width: 2px;`, `border-style: solid;` y `border-color: black;`.

1.4.1. Comparar valores individuales y abreviados

Observad la regla siguiente para el margen (las reglas abreviadas para separación y borde funcionan de la misma manera):

```
div.foo {
  margin-top: 1em;
  margin-right: 1.5em;
  margin-bottom: 2em;
  margin-left: 2.5em;
}
```

Esta regla también se podría escribir:

```
div.foo {
  margin: 1em 1.5em 2em 2.5em;
}
```

1.4.2. Dar menos de cuatro valores para una propiedad abreviada

Un valor abreviado puede incluir menos de cuatro valores, que se aplicarán según se especifica a continuación. Los resultados se ordenan según el número de valores especificados:

1) El mismo valor se aplica a los cuatro lados, por ejemplo `margin: 2px;`.

2) El primer valor se aplica a los márgenes superior e inferior, y el segundo a los márgenes izquierdo y derecho, por ejemplo `margin: 2px 5px;`.

3) Los valores primero y tercero se aplican a los márgenes superior e inferior respectivamente, y el segundo a los márgenes izquierdo y derecho, por ejemplo `margin: 2px 5px 1px;`.

4) Los valores se aplican a los márgenes superior, derecho, inferior e izquierdo respectivamente en el orden en el que aparecen en el CSS, tal como hemos visto antes.

En general, la manera más inteligente de proceder es especificar los cuatro valores en las propiedades abreviadas, por cuestiones de legibilidad. Este consejo también sirve para la propiedad abreviada `padding`.

1.4.3. Elegir entre utilizar una propiedad única o un valor abreviado

Las propiedades `margin` y `padding` abreviadas son las que se utilizan más, aunque hay situaciones en las que es mejor no utilizar las propiedades abreviadas, o como mínimo utilizarlas con mucho cuidado, como por ejemplo las siguientes:

- **Sólo se debe definir un único margen.** En una situación en la que sólo se debe definir una propiedad, el hecho de definir al mismo tiempo múltiples propiedades representa normalmente una violación del principio KISS (*Keep It Simple, Stupid* o No lo compliques, estúpido).

- **El selector al que se aplican las propiedades está sujeto a muchos casos diferentes.** Cuando esto suceda, y tarde o temprano acabará pasando, el inevitable pilón de valores abreviados hará que todo sea difícil de entender cuando debáis reparar o modificar la composición.

- **El mantenimiento de la hoja de estilos que estáis escribiendo irá a cargo de personas con un nivel de habilidades (o de capacidad de razonamiento espacial) deficiente.** Si contáis con que leerán este texto, entonces no es necesario que os preocupéis, pero quizá es mejor no suponer según qué cosas...

- **Debéis cambiar un valor para un caso concreto.** Esto es normalmente consecuencia de un documento o de una hoja de estilos mal diseñada, pero tampoco es una situación excepcional.

1.4.4. Referencia abreviada

1) Definición abreviada de bordes para diferentes propiedades: ya se han explicado al principio de este subapartado. Una cuestión adicional que debemos mencionar es que incluso podéis definir los valores de las propiedades del borde sólo para un único borde del elemento al que se aplica, de la siguiente manera:

```
border-left-width: 2px;
border-left-style: solid;
border-left-color: black;
```

2) Definición abreviada de margen, rellenado y borde para las mismas propiedades: todas actúan de la misma manera.

Podéis ver la definición abreviada de margen, rellenado y borde para las mismas propiedades en el subapartado 1.4.1 de este módulo.

3) Definición abreviada de tipo de letra: podéis especificar el tamaño del tipo de letra, el peso, el estilo, la familia y la interlínea utilizando una definición abreviada como una única línea. Pongamos por caso el CSS siguiente:

```
font-size: 1.5em;
line-height: 200%;
font-weight: bold;
font-style: italic;
font-family: Georgia, "Times New Roman", serif;
```

Podríais especificar exactamente lo mismo con la línea siguiente:

```
font: 1.5em/200% bold italic Georgia, "Times New Roman", serif;
```

4) Definición abreviada de fondo: podéis especificar el color de fondo, la imagen de fondo, la repetición de la imagen y la posición de la imagen con una única línea del CSS. Imaginémonos lo siguiente:

```
background-color: #000;
background-image: url(image.gif);
background-repeat: no-repeat;
background-position: top left;
```

Todo esto mismo se podría representar con la siguiente definición abreviada:

```
background: #000 url(image.gif) no-repeat top left;
```

5) Definición abreviada de lista: aquí también encontramos un código similar con propiedades de una lista que permite definir los valores para el tipo de pico, la posición y la imagen en una única línea. Imaginémonos el siguiente CSS:

```
list-style-type: circle;
list-style-position: inside;
list-style-image: url(bullet.gif);
```

Esto es equivalente a:

```
list-style: circle inside url(bullet.gif);
```

Nota

Observad que `#000` es el valor hexadecimal abreviado para el color; es el equivalente al valor no abreviado #000000 que ya hemos visto antes. Y si queréis ver un ejemplo más complicado, #6c9 es lo mismo que #66cc99.

1.5. Aplicar el CSS al HTML

Hay tres maneras de aplicar el CSS a un documento HTML: estilos en línea, estilos incrustados y hojas de estilos externas. A no ser que tengáis alguna razón muy buena para utilizar uno de los dos primeros métodos, os deberíais decidir siempre por la tercera opción. Pronto veremos el motivo, que es muy obvio, pero antes hablaremos de las diferentes opciones.

1.5.1. Estilos en línea

Podéis aplicar estilos a un elemento utilizando un atributo `style`, de la manera siguiente:

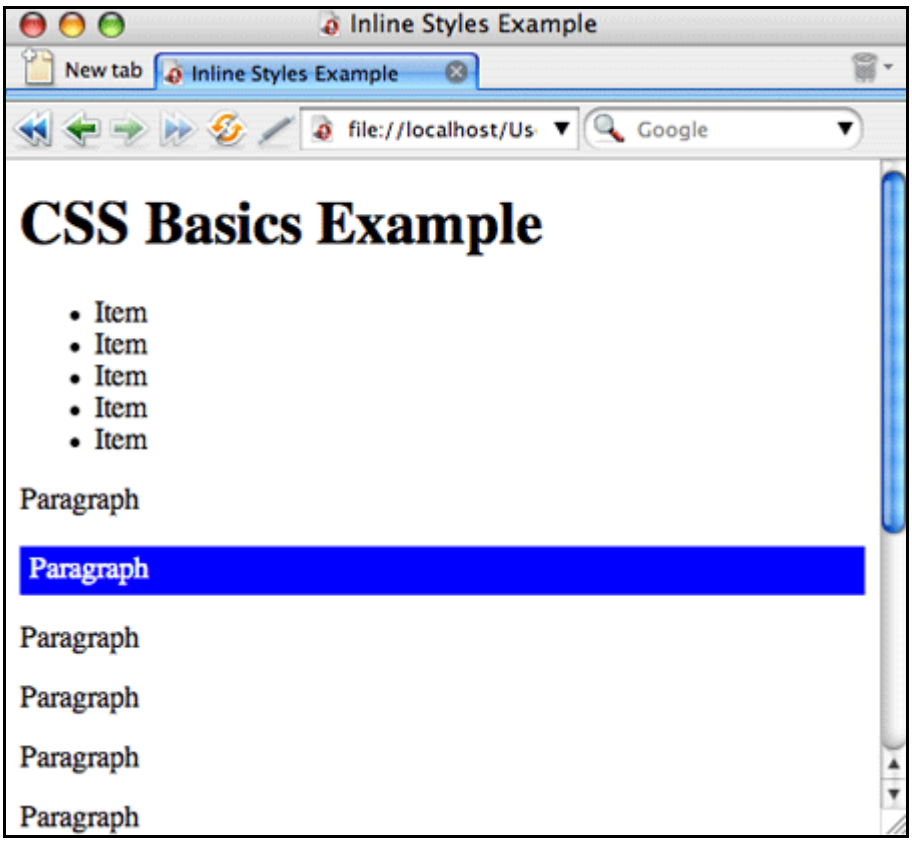
```
<P style="background:blue; color:white; padding:5px;">Paragraph</p>
```

Podéis ver la página ejemplo en: "CSS Basics example"
http://mosaic.uoc.edu/ac/le/operafiles/27/example-inline.html

En este atributo se listan todas las propiedades CSS y sus valores (cada pareja de propiedad/valor se separa de las demás con un signo de punto y coma, y cada propiedad se separa de su valor dentro de la pareja con un signo de dos puntos). Ésta es la manera de definir estilos con CSS.

Si abrís este ejemplo en un navegador y lo miráis, veréis que el párrafo con el atributo `style` es azul con el texto blanco y que tiene un tamaño diferente de los demás, como se puede ver en la figura 1.

Figura 1



Opera muestra el párrafo con los estilos insertados aplicados de una manera diferente a los demás.

La ventaja de los estilos insertados es que el navegador se verá obligado a utilizar estos ajustes. Cualquier otro estilo definido con otras hojas de estilos, o incluso los incrustados en el elemento `head` del documento, quedarán invalidados por estos estilos.

El gran problema de los estilos insertados es que hacen que el mantenimiento sea mucho más difícil de lo que debería ser. El uso de CSS permite separar la presentación del documento de su estructura, mientras que lo que hacen los estilos insertados es precisamente extender reglas de presentación por todo el documento.

Además de la cuestión del mantenimiento, tampoco se aprovecha el aspecto más potente del CSS: la cascada. En el siguiente apartado, volveremos a hablar de la cascada con detalle, pero de momento todo lo que debéis saber es que el uso de la cascada implica definir un aspecto en un lugar que entonces el navegador aplica a todos los elementos que cumplen una regla concreta.

1.5.2. Estilos incrustados

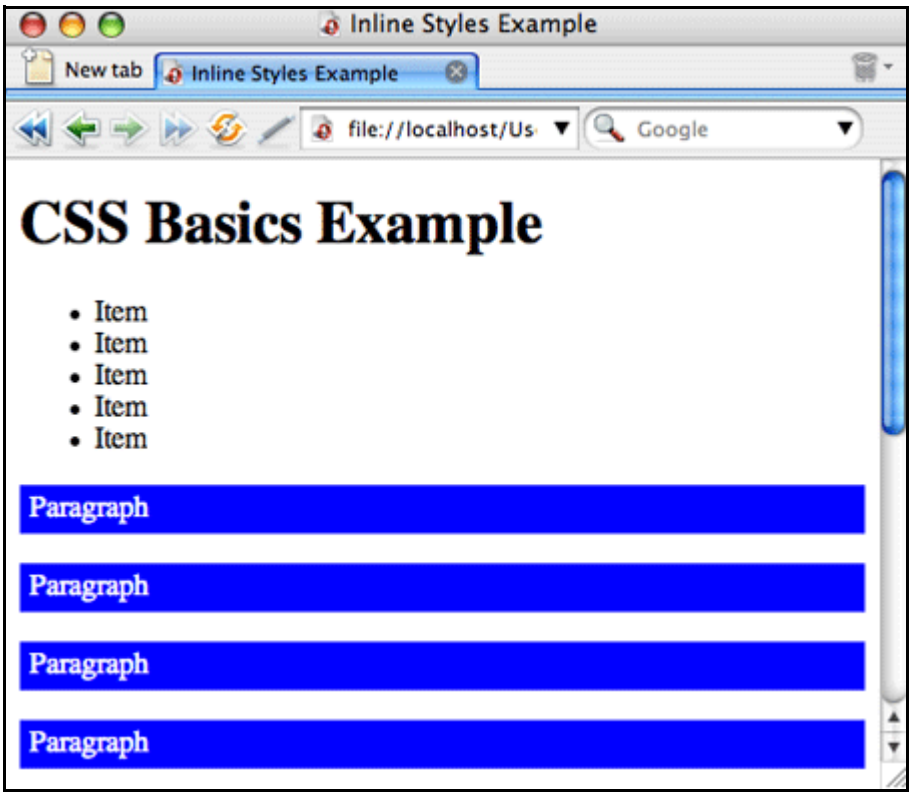
Los estilos incrustados se colocan en el elemento `head` del documento en un elemento `style`, tal como se ve en el ejemplo siguiente:

```
<style type="text/css" media="screen">
  p {
    color:white;
    background:blue;
    padding:5px;
  }
</style>
```

Podéis ver la página ejemplo en: "Embedded example"
http://mosaic.uoc.edu/ac/le/operafiles/27/example-embedded.html

Si abrís el enlace anterior en un navegador, veréis que los estilos definidos se aplican a todos los párrafos del documento, tal como muestra la figura 2. Mirad también el código fuente de la página del ejemplo para ver el CSS del interior de `head`.

Figura 2



Opera muestra todos los párrafos con los estilos definidos en la hoja de estilos incrustados.

Podéis ver la página ejemplo en: "External example"
http://mosaic.uoc.edu/ac/le/operafiles/27/example-external.html

La ventaja de los estilos incrustados es que no hay que añadir un atributo `style` en cada párrafo; los estilos se pueden aplicar a todos los párrafos con

una única definición. Esto también quiere decir que si debéis cambiar el aspecto de todos los párrafos, podréis hacerlo interviniendo en un único lugar, aunque esto está limitado a un documento; pero ¿qué deberíais hacer si quisierais definir al mismo tiempo el aspecto de todos los párrafos de todo un sitio web? Utilizar hojas de estilos externas.

1.5.3. Hojas de estilos externas

Las hojas de estilos externas permiten poner todas las definiciones CSS en un archivo independiente, guardar este archivo con la extensión CSS y entonces aplicarlo a todos los documentos HTML con un elemento `link` en el `head` del documento. Consultad el código fuente de nuestra página de ejemplo*, observad que `head` contiene un elemento `link` que hace referencia a un archivo CSS externo y comprobad que todos los estilos definidos en el archivo CSS** externo se aplican al HTML. Observemos más atentamente este elemento `link`:

*** http://mosaic.uoc.edu/ac/le/operafiles/27/example-external.html**

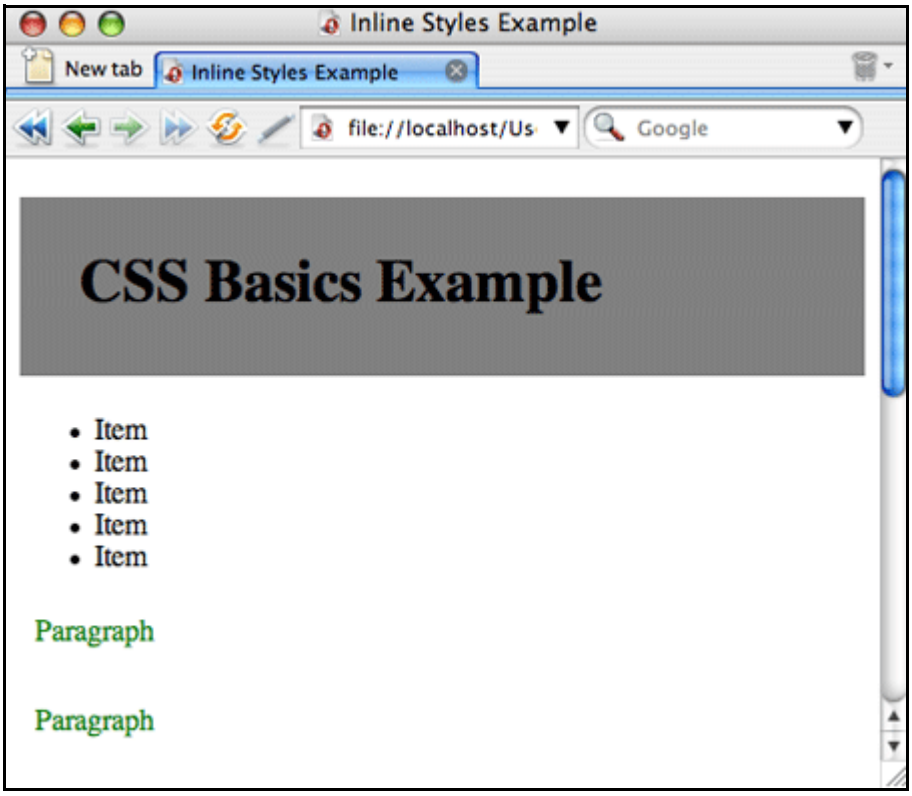
**** http://mosaic.uoc.edu/ac/le/operafiles/27/styles.css**

```
<link rel="stylesheet" href="styles.css" type="text/css" media="screen">
```

En esta asignatura ya hemos hablado del elemento `link`. Sólo para recapitular: el atributo `href` apunta hacia un archivo CSS, el atributo `media` define en qué soportes se aplicarán estos estilos (en este caso, `screen`, ya que no queremos que un documento impreso tenga este aspecto) y `type` define qué es el recurso al que apunta el enlace (una extensión de archivo no es suficiente para determinarlo).

Hemos hablado del elemento link en el apartado 4 del módulo "El texto central de HTML".

Figura 3



Opera muestra los estilos definidos en la hoja de estilos externa cuando se enlaza con un elemento `link`.

Ésta es la mejor situación imaginable: todas las definiciones de aspecto se encuentran en un único archivo, lo que significa que podéis hacer cambios en todo el sitio web sólo cambiando un archivo, y el navegador lo puede cargar una vez y después guardarlo en la memoria caché para todos los demás documentos que hacen referencia a éste, con lo cual se debe descargar una cantidad de datos menor.

1.5.4. Importar hojas de estilos con `@import`

Hay aún otra manera de importar hojas de estilos externas a archivos HTML: la propiedad `@import`. Esta propiedad se inserta en una hoja de estilos incrustada, de la misma manera que el CSS incrustado que hemos visto antes. La sintaxis es la siguiente:

```
<style type="text/css" media="screen">
  @import url("styles.css");
  ... aquí puede haber otros enunciados o estilos CSS para importarlos...
</style>
```

Algunas veces veréis enunciados de importación sin los paréntesis, pero el resultado es el mismo. Otro dato que debéis tener en cuenta es que `@import` debe ocupar siempre el primer lugar en una hoja de estilos incrustada. Finalmente, podéis especificar que la hoja de estilos importada se aplique sólo a algunos tipos de soporte e incluir el tipo de soporte al final del enunciado de importación (esto funciona con todos los navegadores exceptuando IE 6 y versiones anteriores). El siguiente código hace lo mismo que el ejemplo anterior:

```
<style type="text/css">
  @import url("styles.css") screen;
  ... aquí puede haber otros enunciados o estilos CSS para importarlos...
</style>
```

La primera pregunta que os debéis estar haciendo es: "¿por qué necesito otra manera de aplicar hojas de estilos externos a mis documentos HTML"? Pues bien, en realidad no la necesitáis. Sólo incluimos aquí la información sobre `@import` por una cuestión de exhaustividad. El uso de `@import` en lugar de elementos `link` tiene unas cuantas ventajas y desventajas menores, pero éstas son muy menores, por lo cual la decisión será en realidad vuestra. Los elementos `link` son actualmente la manera más aceptada de hacer las cosas:

- Los navegadores antiguos no reconocen `@import` y, por lo tanto, lo ignoran completamente (Netscape 4 y versiones anteriores, así como IE 4 y ver-

siones anteriores si no ponéis el nombre del archivo entre paréntesis). Por lo tanto, podéis utilizar un enunciado `@import` para ocultar estilos a los navegadores antiguos que los utilizarían incorrectamente. Podéis poner los estilos actualizados en una hoja de estilos externa, importarlos con `@import` y ofrecer algunos estilos realmente básicos que no hagan que IE o Netscape 4 se queden encallados en la hoja de estilos incrustada. Os puede ser muy útil, pero actualmente no es muy habitual que sea necesario ofrecer compatibilidad con IE/Netscape 4.

- Como ya hemos comentado, IE 6 no acepta la colocación del tipo de soporte al final de la línea de `@import`; por lo tanto, si queréis insertar múltiples hojas de estilos para diferentes soportes, ésta no es una buena opción.

- Podríais argumentar que el código para múltiples enunciados `@import` es más corto que el código para múltiples elementos `link`, pero esta ventaja es insignificante.

Resumen

En este apartado habéis aprendido a aplicar CSS a documentos HTML, ya sea en forma de estilos insertados utilizando atributos `style`, en forma de estilos incrustados en un elemento `style` en el `head` del documento o como archivos externos en un documento independiente. También habéis aprendido que esta última opción, la de enlazar una hoja de estilos externa utilizando un elemento `link`, es la más recomendable respecto al mantenimiento y al uso de la memoria caché. Finalmente, hemos hablado sobre la sintaxis básica del CSS y hemos explicado los comentarios, diferentes tipos de selectores y la agrupación de selectores.

En el siguiente apartado seguiremos viendo detalles de CSS y hablaremos con detenimiento de la cascada y de la especificidad de los selectores.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿Cuáles son las ventajas y los problemas de los estilos en línea y cómo los aplicáis a un elemento?

2. ¿Qué es una regla de estilo? Describid sus diferentes componentes y su sintaxis.

3. Imaginad que tenéis unas cuantas reglas de estilos; ¿qué debéis hacer para convertirlas en una hoja de estilos externa?

4. ¿Con qué se corresponde el siguiente selector CSS: `ul#nav{}`?

5. ¿Cuál es la ventaja de agrupar selectores?

6. ¿Cómo podéis definir una hoja de estilos de impresión?

2. Herencia y cascada

Tommy Olsson

La herencia y la cascada son dos conceptos básicos en CSS. Se deben comprender bien para utilizar CSS. Por suerte, no son muy difíciles de entender, aunque algunos detalles pueden ser un tanto complicados de recordar.

Ambos conceptos están relacionados, pero son diferentes. La herencia está relacionada con cómo los elementos del etiquetado de HTML heredan propiedades de sus elementos padres (los que los contienen) y los transmiten a sus hijos, mientras que la cascada tiene que ver con las declaraciones de CSS que se aplican a un documento y cómo las reglas contradictorias se anulan o no entre ellas.

En este apartado, trataremos detalladamente estos dos conceptos. Probablemente, la herencia es un concepto más fácil de captar, de manera que empezaremos con éste y después pasaremos a las particularidades de la cascada.

Nota

Descargad el código fuente de los ejemplos de este apartado; el fichero "inheritance_cascade_code.zip" contiene el CSS y HTML acabados, además de la plantilla inicial de HTML para que podáis ir trabajando mientras veis los ejemplos.

Descargad los ejemplos en: "inheritance_cascade_code.zip"
http://mosaic.uoc.edu/ac/le/operafiles/28/inheritance_cascade_code.zip

2.1. Herencia

La herencia en CSS es el mecanismo mediante el cual determinadas propiedades de un elemento padre se transmiten a sus hijos. De hecho, se parece mucho a la herencia genética. Si los progenitores tienen los ojos azules, los hijos seguramente también tendrán los ojos azules.

No todas las propiedades CSS son heredadas, porque algunas de ellas no tendría sentido que lo fueran. Por ejemplo, los márgenes no se heredan porque es poco probable que un elemento hijo necesite los mismos márgenes que su padre. Normalmente, el sentido común dicta qué propiedades se heredan y cuáles no, pero para estar del todo seguros, debemos consultar cada propiedad en la tabla de resumen de propiedades de la especificación CSS 2.1*.

* http://www.w3.org/TR/CSS21/propidx.html

2.1.1. Para qué sirve la herencia

¿Por qué tiene CSS un mecanismo de herencia? Probablemente, la manera más sencilla de responder a esta pregunta sea pensar qué pasaría si no existiera la herencia. Se deberían especificar cuestiones como la familia de fuentes, el tamaño de la fuente y el color del texto individualmente para todos y cada uno de los tipos de elemento.

Mediante la herencia, por ejemplo, se pueden especificar las propiedades de las fuentes de los elementos `html` o `body` y todo el resto de elementos los heredarán. Se pueden especificar los colores de fondo y de primer plano de un elemento contenedor concreto y todos los elementos hijos de este contenedor heredarán automáticamente el color de primer plano. El color de fondo no se hereda, pero el valor inicial de `background-color` (color de fondo) es `transparent`, lo cual significa que el fondo del padre se verá a través de él. El efecto es el mismo que si se heredara el color de fondo, pero pensad qué sucedería si se heredaran las imágenes de fondo: todos los hijos tendrían la misma imagen de fondo como padre y, por lo tanto, todo parecería un rompecabezas creado por alguien con problemas de drogas, ya que el fondo "volvería a empezar desde cero" para cada elemento.

2.1.2. Cómo funciona la herencia

Todos los elementos de un documento HTML heredan todas las propiedades heredables de su padre excepto el elemento raíz (`html`), que no tiene progenitor.

El hecho de que las propiedades heredadas tengan algún efecto o no depende de otros factores, como veremos más adelante cuando hablemos de la cascada. De la misma manera que una madre de ojos azules puede tener un hijo de ojos marrones si el padre tiene los ojos marrones, las propiedades heredadas en CSS pueden anularse.

2.1.3. Un ejemplo de herencia

1) Copiad el siguiente documento HTML en un fichero nuevo del editor de textos que más os guste y guardadlo como `inherit.html`.

```
<!DOCTYPE html>
  <head>
    <meta charset=utf-8”>
    <title>Herencia</title>
  </head>
  <body>
    <h1>Título</h1>
```

```
    <p>Un párrafo de texto.</p>
  </body>
</html>
```

Si abrís el documento en el navegador web, veréis un documento bastante aburrido que se muestra según el estilo por defecto de vuestro navegador.

2) Cread un nuevo fichero vacío con el editor de textos, copiad dentro la regla CSS que se muestra a continuación y guardad el fichero como `style.css` en la misma ubicación que el fichero HTML.

```
html {
  font: 75% Verdana,sans-serif;
}
```

3) Enlazad la hoja de estilos en el documento HTML insertando la línea siguiente antes del tag `</head>`.

```
<link rel="stylesheet" type="text/css" media="screen" href="styles.css">
```

4) Guardad el fichero HTML modificado y recargad el documento en el navegador. La fuente pasará de ser la predeterminada por el navegador (normalmente Times o Times New Roman) a ser Verdana. Si no tenéis Verdana instalada en el ordenador, el texto se mostrará con la fuente Sans Serif especificada por defecto en la configuración del navegador. Además, el texto se verá un 25% más pequeño que en la versión sin estilo.

La regla CSS que hemos especificado se aplica únicamente al elemento `html`. No hemos especificado ninguna regla para los títulos o los párrafos, pero ahora todo el texto se muestra en Verdana al 75% del tamaño por defecto. ¿Por qué? Por la herencia.

La propiedad `font` es una propiedad abreviada que establece toda una serie de propiedades relacionadas con las fuentes. Sólo hemos especificado dos, el tamaño de la fuente y la familia de fuentes, pero esta regla equivale a lo siguiente:

```
html {
  font-style: normal;
  font-variant: normal;
  font-weight: normal;
```

```
  font-size: 75%;
  line-height: normal;
  font-family: Verdana,sans-serif;
}
```

Todas estas propiedades se heredan, de manera que el elemento `body` las heredará del elemento `html` y después las transmitirá a sus hijos: el título y el párrafo.

Pero, ¡un momento! Hay algo que no acaba de quedar claro respecto a la herencia del tamaño de la fuente, ¿verdad? El tamaño de la fuente del elemento `html` se establece en 75%, pero ¿75% de qué? ¿Y el tamaño de la fuente de `body` no debería ser el 75% del tamaño de la fuente de su padre y los tamaños de las fuentes del título y del párrafo deberían ser el 75% del tamaño del elemento `body`?

El valor que se hereda no es el valor especificado, es decir, el valor que escribimos en la hoja de estilo, sino algo que se llama *el valor computado*. El valor computado es, en el caso del tamaño de la fuente, un valor absoluto medido en píxeles. Para el elemento `html`, que no tiene un elemento padre del cual heredar, un porcentaje del valor de tamaño de fuente se asocia al tamaño de fuente predeterminada del navegador. La mayoría de los navegadores actuales tienen un tamaño de fuente predeterminada de 16 píxeles. El 75% de 16 son 12, de manera que el valor computado del tamaño de la fuente del elemento `html` será probablemente 12 píxeles. Éste es el valor que hereda `body` y que se transmite al título y al párrafo.

(El tamaño de la fuente del título es mayor porque el navegador aplica algunas normas de estilo integradas propias. Podéis ver el tema de la cascada a continuación.)

5) Añadid dos declaraciones más a la regla de la hoja de estilo de CSS:

```
html {
  font: 75% Verdana,sans-serif;
  background-color: blue;
  color: white;
}
```

6) Guardad el fichero CSS y recargad el documento en el navegador.

Ahora el fondo es de color azul fuerte y todo el texto es blanco. La regla se aplica al elemento `html`: el documento entero, cuyo fondo será azul. El elemento `body` hereda el color blanco de primer plano y se transmite a todos los hijos de `body`: en este caso, el título y el párrafo. Éstos no heredan el fondo, pero el fondo se establecerá en el valor por defecto de `transparent`, de manera que el resultado visual final será texto blanco sobre fondo azul.

7) Añadid otra regla nueva a la hoja de estilo y guardad y recargad el documento.

```
h1 {
  font-size: 300%;
}
```

Esta regla establece el tamaño de la fuente del título. El porcentaje se aplica al tamaño de fuente heredada (el 75% de la predeterminada por el navegador, que suponemos que es 12 píxeles), de manera que el tamaño del título será el 300% de 12 píxeles, es decir: 36 píxeles.

2.1.4. Forzar la herencia

Mediante la palabra clave `inherit` (heredar) puede forzarse la herencia incluso para propiedades que no se heredan normalmente. Sin embargo, se debe utilizar con mucho cuidado porque Microsoft Internet Explorer (hasta la versión 7 incluida) no es compatible con esta palabra clave.

La regla siguiente hace que todos los párrafos hereden todas las propiedades de fondo de sus padres:

```
p {
  background: inherit;
}
```

Con las propiedades abreviadas se puede utilizar `inherit` en vez de los valores normales. Se debe utilizar la versión abreviada o bien para todo o bien para nada en absoluto. En la versión no abreviada no se pueden especificar algunos valores y utilizar `inherit` para otros porque los valores pueden darse en cualquier orden y no hay manera de especificar qué valores queremos heredar.

Forzar la herencia no es algo que haya que hacer a menudo. Puede ser útil para "deshacer" una declaración de una regla conflictiva o para no tener que introducir los datos de determinados valores directamente en el código fuente. Un ejemplo de esto sería el típico menú de navegación:

```
<ul id="nav">
  <li><a href="/">Inicio</a></li>
  <li><a href="/news/">Noticias</a></li>
  <li><a href="/products/">Productos</a></li>
```

```
  <li><a href="/services/">Servicios</a></li>
  <li><a href="/about/">Sobre nosotros</a></li>
</ul>
```

Para mostrar esta lista de enlaces como menú horizontal, podéis utilizar el CSS siguiente:

```
#nav {
  background: blue;
  color: white;
  margin: 0;
  padding: 0;
}
#nav li {
  display: inline;
  margin: 0;
  padding: 0 0.5em;
  border-right: 1px solid;
}
#nav li a {
  color: inherit;
  text-decoration: none;
}
```

En este caso, el color de fondo de toda la lista se establece en azul en la regla de `#nav`. Así, también se establece el color de primer plano como blanco y todos los elementos de la lista y todos los enlaces heredan el mismo. La regla de los elementos de la lista establece un límite a la derecha, pero no especifica el color del margen, lo que significa que utilizará el color de primer plano heredado (el blanco). Para los enlaces, hemos utilizado `color:inherit` para forzar la herencia y anular el color de los enlaces predeterminado del navegador.

Lógicamente, también podría haber especificado el blanco como color del margen y del texto de los enlaces, pero lo mejor del hecho de dejar que lo haga la herencia es que, si más adelante decidimos cambiar los colores, sólo deberemos hacer un cambio en este punto.

2.2. Cascada

CSS significa *cascading style sheets* (hojas de estilo en cascada) y, por lo tanto, no debería extrañarnos que la cascada sea un concepto importante. Es el mecanismo que controla el resultado final cuando se aplican varias declaraciones CSS contrapuestas al mismo elemento.

Hay tres conceptos principales que controlan el orden en el que se aplican las declaraciones de CSS:

1) Importancia.
2) Especificidad.
3) Orden en las fuentes.

A continuación, trataremos con detalle estos conceptos uno a uno.

La importancia es uno de los conceptos más... pues... importantes. Si dos declaraciones tienen la misma importancia, la especificidad de las reglas decidirá cuál se debe aplicar. Si las reglas tienen la misma especificidad, el orden de las fuentes controla el resultado.

2.2.1. Importancia

La importancia de una declaración de CSS depende de dónde se ha especificado. Las declaraciones contrapuestas se aplicarán en el orden siguiente: las nuevas anularán a las más antiguas.

1) Hoja de estilos de agente de usuario.
2) Declaraciones normales en hojas de estilo de autor.
3) Declaraciones normales en hojas de estilo de usuario.
4) Declaraciones importantes en hojas de estilo de autor.
5) Declaraciones importantes en hojas de estilo de usuario.

Una **hoja de estilos de agente de usuario** es la hoja de estilo integrada del navegador. Cada navegador tiene sus propias reglas sobre cómo mostrar varios elementos de HTML si el usuario o diseñador de la página no especifica ningún estilo. Por ejemplo, los enlaces no visitados suelen ser azules y estar subrayados.

Una **hoja de estilos de usuario** es una hoja de estilo que ha especificado el usuario. No todos los navegadores son compatibles con las hojas de estilo de usuario, pero pueden ser muy prácticas, sobre todo para los usuarios con determinados tipos de minusvalía. Por ejemplo, una persona disléxica puede tener una hoja de estilo de usuario que especifique determinadas fuentes y colores que le faciliten la lectura.

Opera permite especificar hojas de estilo de usuario desde `Tools` (herramientas) (o desde el menú `Opera` en un Mac) > `Preferences...` (preferencias) > pestaña `Advanced` (avanzado) > `Content` (contenido), haciendo clic en el botón `Style Options...` (opciones de estilo...) y después señalando la hoja de estilo de usuario del campo de texto `My style sheet` (mi hoja de estilo)

dentro de la pestaña `Display` (mostrar) de este cuadro de diálogo. También se puede especificar si se quiere que la hoja de estilos de usuario anule la hoja de estilos del autor (el diseñador de la web) en la pestaña `Presentation` (presentación) e, incluso, que añada un botón a la interfaz de usuario con el que poder cambiar la hoja de estilo del usuario con la del autor. Para hacerlo, salid totalmente del menú `Preferences...` (preferencias...) haciendo clic en `Aceptar` y después haced clic con el botón derecho o, mientras apretáis `Ctrl`, haced clic en algún punto de la interfaz del navegador Opera, seleccionad la vista `Customize...` (personalizar...) > pestaña `Buttons` (botones) > `Browser` (navegador) y arrastrad el botón `Author Mode` (modo autor) hasta un punto de alguna de las barras de herramientas.

Cuando hablamos de "hojas de estilo", normalmente hacemos referencia a una hoja de estilo de autor. Es la hoja de estilos que ha creado o enlazado el autor del documento (o, más probablemente, el diseñador de la web).

Las declaraciones normales son exactamente lo que su nombre indica: declaraciones normales. Lo contrario son las **declaraciones importantes**, que son las declaraciones que van seguidas de una directiva `!important`.

Como se puede observar, las declaraciones importantes de una hoja de estilo de usuario tienen prioridad sobre todas las demás, lo cual es lógico. Siguiendo el ejemplo de la persona disléxica, podría ser que quisiera ver todo el texto con Comic Sans MS en caso de que le facilitara la lectura. Entonces podría tener una hoja de estilo de usuario con la regla siguiente:

```
* {
  font-family: "Comic Sans MS" !important;
}
```

En este caso, independientemente de lo que haya especificado el diseñador, e independientemente de aquello que se haya establecido como familia de fuentes predeterminada del navegador, todo se mostrará con Comic Sans MS.

El método de presentación por defecto del navegador sólo se aplicará si las declaraciones no quedan anuladas por alguna regla de una hoja de estilo de usuario o una hoja de estilo de autor, ya que la hoja de estilo de agente de usuario tiene precedencia menor.

En realidad, la mayoría de los diseñadores no se deben preocupar demasiado de la importancia porque no se puede hacer nada al respecto. No hay ninguna manera de saber si un usuario tiene una hoja de estilos de usuario definida que anule nuestro CSS. Y si la tiene, seguramente sea por alguna buena razón. Aun así, es útil saber qué es la importancia y cómo puede afectar a la presentación de nuestros documentos.

2.2.2. Especificidad

La especificidad es algo que todos los autores de CSS deben comprender y tener en cuenta. Puede considerarse una medida de cuán específico es el selector de una regla. Un selector de especificidad baja puede dar como resultado muchos elementos (como `*`, que da como resultado todos los elementos del documento), mientras que un selector con una especificidad elevada puede que sólo dé como resultado un único elemento de una página (como `#nav`, que sólo da como resultado el elemento con una `id` de `nav`).

La especificidad de un selector puede calcularse fácilmente, como veremos muy pronto. Si dos o más declaraciones entran en conflicto por un elemento determinado y todas las declaraciones tienen la misma importancia, la de la regla con el selector más específico será la que "gane".

La **especificidad** tiene cuatro componentes; por ejemplo a, b, c y d. El componente "a" es el más distintivo y el "d", el que menos.

- El componente "a" es bastante sencillo: es 1 para una declaración en un atributo `style`; si no, es 0.

- El componente "b" es el número de selectores de `id` en el selector (los que empiezan con `#`).

- El componente "c" es el número de selectores de atributo, incluidos los selectores de clase y pseudoclases.

- El componente "d" es el número de tipo de elementos y pseudoelementos del selector.

Así, después de contar un poco, podemos unir estos cuatro componentes para conseguir la especificidad de cualquier regla. Las declaraciones de CSS en un atributo `style` no tienen selector, de manera que su especificidad siempre es 1,0,0,0.

Veamos unos cuantos ejemplos que nos ayudarán a aclarar cómo funciona este proceso.

Selector	a	b	c	d	Especificidad
`h1`	0	0	0	1	0,0,0,1
`.foo`	0	0	1	0	0,0,1,0
`#bar`	0	1	0	0	0,1,0,0
`html>head+body ul#nav *.home a:link`	0	1	2	5	0,1,2,5

Pasemos ahora a comentar el último ejemplo con más detalle. Se obtiene `a=0` porque es un selector, no una declaración de un atributo `style`. Hay un selector ID

(`#nav`), de manera que `b=1`. Hay un selector de atributos (`.home`) y una pseudo-clase (`:link`), por lo tanto, `c=2`. Hay cinco tipos de elemento (`html`, `head`, `body`, `ul` y `a`), de manera que `d=5`. Por lo tanto, la especificidad final es 0,1,2,5.

Hay que mencionar que los combinadores (como `>`, `+` y el espacio en blanco) no afectan a la especificidad de un selector. El selector universal (`*`) tampoco cuenta para calcular la especificidad.

También hay que tener en cuenta que existe una gran diferencia de especificidad entre un selector `id` y un selector de atributos que por casualidad haga referencia a un atributo `id`. Aunque den como resultado el mismo elemento, tienen especificidades muy diferentes. La especificidad de `#nav` es 0,1,0,0 y la especificidad de `[id="nav"]` es sólo 0,0,1,0.

Fijémonos en cómo funciona esto en la práctica.

1) Empezad añadiendo otro párrafo al documento HTML.

```
<body>
  <h1>Título</h1>
  <p>Un párrafo de texto.</p>
  <p>Un segundo párrafo de texto.</p>
</body>
```

2) Añadid una regla a la hoja de estilo para hacer que el texto de los párrafos sea de un color diferente.

```
p {
  color: cyan;
}
```

3) Guardad los dos ficheros y recargad el documento en vuestro navegador. Se deberían ver dos párrafos de color cian.

4) Estableced una `id` en el primer párrafo para que podáis apuntar hacia él fácilmente con un selector CSS.

```
<body>
  <h1>Título</h1>
  <p id="special">Un párrafo de texto.</p>
  <p>Un segundo párrafo de texto.</p>
</body>
```

5) Añadid una regla para el párrafo especial a la hoja de estilo.

```
#special {
  background-color: red;
  color: yellow;
}
```

6) Guardad los dos ficheros y recargad el documento en el navegador para ver el resultado, que es bastante colorido.

Veamos las declaraciones que se aplican en los dos párrafos.

La primera regla que hemos añadido establece `color:cyan` para todos los párrafos. La segunda regla establece un color de fondo rojo y establece `color:yellow` para el único elemento que tiene `id` de `special`. El primer párrafo encaja con estas dos reglas; es un párrafo y tiene la `id` de `special`.

El fondo rojo no es ningún problema porque sólo se ha especificado para `#special`. No obstante, ambas reglas incluyen una declaración de la propiedad de `color`, lo que significa que hay un conflicto. Ambas reglas tienen la misma importancia, se trata de declaraciones normales en la hoja de estilos de autor, de manera que hay que fijarse en la especificidad de los dos selectores.

El selector de la primera regla es `p`, que tiene una especificidad de 0,0,0,1 según las reglas anteriormente mencionadas, ya que incluye un único tipo de elemento. El selector de la segunda regla es `#special`, que tiene una especificidad de 0,1,0,0 porque está formado por un selector de `id`. 0,1,0,0 y es mucho más específico que 0,0,0,1, de manera que la declaración `color:yellow` gana y se obtiene el texto amarillo sobre fondo rojo.

2.2.3. Orden en las fuentes

Si dos declaraciones afectan al mismo elemento, tienen la misma importancia y la misma especificidad, la señal distintiva final es el orden en las fuentes. La declaración que se ve más adelante en las hojas de estilo "ganará" a las anteriores.

Si tenéis una única hoja de estilo externa, las declaraciones al final del fichero anularán a las que sucedan antes al fichero en caso de conflicto. Las declaraciones contrapuestas también pueden suceder en diferentes hojas de estilo. En este caso, el orden en el que se enlazan, se incluyen o se importan las hojas de estilo determina qué declaración se aplica, de manera que si se tienen dos hojas de estilo enlazadas en el `head` de un documento , la enlazada al último anulará a la enlazada al primero. Veamos un ejemplo práctico de cómo funciona esto.

1) Añadid una regla nueva a la hoja de estilo, justo al final del fichero, como por ejemplo:

```
p {
  background-color: yellow;
  color: black;
}
```

2) Guardad y recargad la página web. Ahora tenéis dos reglas que dan como resultado todos los párrafos. Tienen la misma importancia y la misma especificidad (ya que el selector es el mismo); por lo tanto, el mecanismo final para distinguirlas será el orden de las fuentes.

La última regla especifica `color:black` y anulará a `color:cyan` de la regla anterior.

Fijaos en cómo esta regla nueva no afecta en absoluto al primer párrafo. Aunque la regla nueva aparece en último lugar, su selector tiene una especificidad más baja que la de `#special`. Esto demuestra claramente cómo la especificidad tiene prioridad sobre el orden de las fuentes.

Resumen

Herencia y cascada son conceptos básicos que cualquier diseñador web debe comprender.

La herencia permite declarar propiedades en elementos de nivel alto y que estas propiedades se transmitan a todos los elementos descendientes. Sólo algunas propiedades se heredan por defecto, pero la herencia puede forzarse mediante la palabra clave `inherit`.

La cascada soluciona los conflictos cuando varias declaraciones afectan a un elemento determinado. Las declaraciones importantes anulan a las que no lo son tanto. Entre declaraciones de igual importancia, la especificidad de la regla controla cuál se aplica. Y, si todas las demás son iguales, el orden de las fuentes supone la distinción definitiva.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿Se puede heredar la propiedad `width`? Pensad en ello antes de contestar (¿tendría sentido?) y después mirad la respuesta correcta en la especificación CSS*.

* http://www.w3.org/TR/CSS21

2. Si añadimos `!important` a la declaración `color:black` de la última regla de la hoja de estilo de ejemplo*, ¿afectará al color del texto del primer párrafo "especial"?

*** http://mosaic.uoc.edu/ac/le/operafiles/28/inheritance_cascade_code.zip**

3. ¿Qué selector es más específico, "`#special`" o "`html>head+body>h1+p`"?

4. ¿Qué apariencia debería tener una hoja de estilo de usuario para que nuestro documento de prueba aparezca con Comic Sans MS negra sobre fondo blanco, independientemente de la hoja de estilo del autor?

3. Dar estilos al texto con CSS

Ben Henick

Como la web es una colección de documentos, algunos dinámicos, otros estáticos y otros funcionales, las convenciones con las que se les aplica un formato se extraen de nuestro mejor punto de referencia: los seis siglos de historia de la imprenta. Y esto incluye la tipografía. Sin embargo, la web es un medio nuevo y diferente, y la tipografía para sitios web necesita un conjunto de competencias de diseño editorial muy diferente y, además, está mucho más limitada.

Este apartado se basa en los conocimientos adquiridos previamente en la asignatura y ofrece una guía detallada para aplicar estilo a un texto de manera efectiva utilizando CSS.

Nota

A continuación, se enlaza con varios ejemplos que demuestran las técnicas mencionadas; podéis descargaros los 29 ejemplos del apartado con el fichero "article29_examples.zip".

Descargad los ejemplos en: "article29_examples.zip"
http://mosaic.uoc.edu/ac/le/operafiles/29/article29_examples.zip

3.1. Repaso de tipografía en la web

En la web, los diseñadores tienen mucho menos control sobre la presentación que en otros medios. Este hecho se ve claramente cuando se trata de las propiedades del lienzo del documento como el tamaño, la resolución y el contraste. También hay muchas limitaciones con respecto a la calidad de la tipografía de la web, un tema que ya se ha tratado.

Podéis ver los conceptos básicos de tipografía en el apartado 6 del módulo "Conceptos de diseño web".

Otros aspectos que merecen ser tratados son el contraste y la facilidad de lectura, de los cuales hablaremos a continuación.

3.1.1. Contraste

El **contraste** del tipo de letra, la facilidad con la que pueden distinguirse las líneas del espacio en blanco y de las líneas adyacentes, depende de varios factores como la luminosidad, el color, el tamaño y la composición. Se menciona en este apartado por el mero hecho de destacar que el texto de contraste bajo se debería establecer en el mayor tamaño posible.

3.1.2. Facilidad de lectura

En un contexto de diseño utilizamos la palabra inglesa **legibility** para referirnos a la facilidad con la que podemos buscar fragmentos específicos de información, mientras que con su sinónimo **readabilty** nos referimos a la facilidad de comprensión de un texto. Las decisiones de diseño que potencian una u otra calidad se enumeran en la tabla 1.

*** http://en.wikipedia.org/wiki/Leading**

Tabla 1. Características de la facilidad de lectura

Objetivo	Longitud de línea	Canales y espacio interlineal*	Tipo	Justificado
'Readability'	moderada	aumentados	serif	inexistente [izquierda]
'Legibility'	corta	normales	sans-serif	variable, a menudo completo

En el próximo apartado, "El modelo de composición de CSS", se hablará sobre la anchura óptima de columna.

3.2. Propiedades CSS de fuentes: cambiar el aspecto del texto

La composición de textos es la manipulación del texto con respecto a la composición general y al aspecto de las letras y las palabras. Esta segunda tarea se lleva a cabo mediante las propiedades CSS de fuente, de las que hablaremos a continuación.

3.2.1. `font-size` y selección del tipo de unidad

Como los documentos generalmente varían de tamaño de fuente más que de tipo de letra y las variantes de fuentes suelen manipularse con las hojas de estilos de usuario, la propiedad principal de interés es `font-size` (tamaño de la fuente). Cuando se utiliza en una regla, va seguida de un valor que especifica una medida de unidad o, en ocasiones, un tamaño de palabra clave (como pequeño o medio).

Cómo se hace

La declaración de `font-size` más importante de una hoja de estilo es más o menos así:

```
body {...
  font-size: 14px;
  …}
```

La herencia hace que todas las especificaciones de tamaño de la fuente de un documento se basen en el tamaño declarado por el documento `body`, ya sea en la hoja de estilo del navegador o en la nuestra.

El valor por defecto típico de 16 píxeles de los navegadores es un buen punto de inicio para la parte central del texto, pero la mayoría de los visitantes pueden leer sin problemas tipos de letra más pequeños. Por lo tanto, la mayoría de los diseñadores eligen un tamaño más pequeño, de unos 11-14 píxeles.

La herencia se aplica al tamaño del tipo de letra cuando se especifica un tamaño relativo y cuando se especifica un tamaño de palabra clave para un elemento con un progenitor cuyo tamaño no se especifica con una palabra clave.

3.2.2. ¿Qué tipos de unidad pueden aplicarse en las propiedades de texto CSS?

En las reglas de hojas de estilo, los tipos de unidad que suelen aplicarse más a menudo al texto son píxeles (`px`), ems (`em`, se explica más adelante), porcentajes (`%`) y puntos (`pt`). En la tabla 2 se describe el comportamiento del texto redimensionado con estas unidades.

Tabla 2. Unidades de CSS convenientes para dimensionar un texto

Unidad	Definición[1]	Uso
Ems de CSS	$\Delta = x$	`1,333 em`
Palabras clave	Definido por el agente de usuario[2]	`large`, *etc.*
Porcentaje	$\Delta = x \div 100$	`133.3%`
Píxeles	Unidad absoluta	`16 px`
Puntos	Unidad absoluta	`12 pt`

1 Δ es la **proporción** de cambio en el tamaño de la fuente a partir del valor heredado.
2 Sólo se hereda el valor del tamaño no especificado en palabra clave más cercano.

Otros tipos de unidad posibles son las palabras clave de tamaño, las picas (`pc`, una pica equivale exactamente a 12 puntos) y las exes (`ex`). También están disponibles la mayoría de los otros tipos de unidad que acepta CSS2, pero no se utilizan casi nunca cuando se trabaja con texto.

3.2.3. ¿Para qué sirve tener tantos tipos de unidad diferentes?

Como se indica en la tabla 2, hay unidades de tamaño relativo y absoluto. Las palabras clave incluyen ambas características, es decir, tamaño absoluto entre sí, pero relativas al valor no especificado en palabra clave que heredan. La mejor manera de utilizarlas es ésta:

- Los tamaños **absolutos** (`px`, unidades estandarizadas como `pt`) es mejor aplicarlas a composiciones que no cambien en relación con las propiedades de lienzo del documento, las llamadas composiciones “fijas”, “estáticas” o “de hielo”.

- Los tamaños **relativos** (`em`, `%`) deberían utilizarse en composiciones no estáticas y en situaciones en las que hay que establecer un compromiso entre la usabilidad del sitio y el control del diseñador de la composición.

- Los tamaños de **palabra clave** (que se explican a continuación) deberían utilizarse cuando la usabilidad prevalezca por encima de todas las demás consideraciones del diseño.

3.2.4. Tamaños absolutos y usabilidad

Las versiones antiguas de Internet Explorer no permiten al visitante modificar el tamaño del texto que se ha establecido en tamaños absolutos y las interfaces de modificación del tamaño del texto de algunos navegadores que sí que permiten este nivel de control de usuario pueden costar de encontrar (los usuarios de Opera y Firefox lo tienen fácil con Shift + Ctrl/Cmd + más/menos y Ctrl/Cmd + más/menos, respectivamente). A causa de estas limitaciones, se suele establecer el `font-size` de `body` en un valor relativo, normalmente en unidades em, que se supone que el navegador controla por defecto.

3.2.5. ¿Cuál es el equivalente físico de un píxel?

La respuesta estrictamente correcta es que no existe. Los píxeles son simplemente una unidad de resolución de hardware de visualización. No obstante...

Aunque es estrictamente imposible definir o predecir las dimensiones literales de un píxel, los patrocinadores de proyectos comerciales más nerviosos suelen sorprenderse cuando descubren que el diseño de su sitio adquiere un aspecto diferente en los ordenadores de los clientes que tienen una configuración diferente de la suya y riñen al diseñador web por este motivo. Por lo tanto, puede ser útil comprender cómo funcionan los píxeles. Esta información os servirá para cuando alguien que os haya encargado crear una página web se queje de que el texto no queda exactamente igual en diferentes ordenadores.

Las empresas de software mantienen un acuerdo informal según el cual, 96 ppp (píxeles por pulgada) es un tamaño razonable. De esta manera, un texto central de 16 píxeles se imprimirá a un tamaño de una sexta parte de una pulgada o 12 puntos. En la pantalla de cristal líquido cada vez más habitual de 17" 1280x800, un texto de 16 píxeles tendrá un tamaño aproximado de 13 puntos en la pantalla, pero en una pantalla parecida de ordenador portátil de 15", el tamaño será de 11,44 puntos.

Estas medidas son efectivas con la configuración por defecto. La mayoría de los entornos permiten al usuario final establecer una medida habitual de ppp, de manera que siempre se darán casos diferentes.

En conclusión: un píxel es un píxel, pero todo lo demás es variable.

3.2.6. Ems, porcentajes y puntos según el CSS

Tradicionalmente, un em es equivalente a la altura de la "M" mayúscula de la fuente de que hablemos. Sin embargo, en CSS, la unidad em es el equivalente a la altura total de una línea; o, dicho de otro modo, para un elemento al que se le ha establecido el `font-size` a 14 píxeles:

```
1em = 100% = 14px
```

En entornos normales, esta declaración puede ampliarse a:

```
1em = 100% = 14px = 10,5pt
```

El aumento y la reducción del tamaño funcionan de manera multiplicativa, de modo que si hay un segundo elemento que se quiera definir con un tamaño de 16 píxeles, teniendo en cuenta la herencia habitual, todo lo que se muestra a continuación obtendría el resultado deseado:

```
1,143em = 114,3% ≈ 16px = 12pt
```

3.2.7. Una breve nota sobre la recomendación oficial CSS 2.1

En ocasiones, os dirán que consultéis la Recomendación Candidata o *Candidate Recommendation* del World Wide Web Consortium para la especificación CSS. Como todas las recomendaciones del W3C*, este documento puede considerarse la definición de un estándar web; algunos se siguen más que otros, tanto por parte de los fabricantes de navegadores como por los desarrolladores de webs.

* http://www.w3.org/TR/css-2010/

Aunque es útil conocer de arriba abajo las recomendaciones del W3C, esta asignatura se ha diseñado para proporcionar una introducción concisa y fácil de asimilar al diseño/desarrollo web y las recomendaciones del W3C pueden ser un poco pesadas. En todos los casos en los que se os pida visitar la recomendación de CSS 2.1, se os está dirigiendo a un material demasiado complejo como para justificar una explicación detallada en este apartado.

3.2.8. Palabras clave de tamaño

También se pueden utilizar palabras clave de tamaño, como se ha mencionado anteriormente. Hay siete, desde `xx-small` hasta `xx-large`, y `medium` es el valor medio (y predeterminado). La lista completa de los siete valores se muestra más adelante en la tabla 3, que incluye todas las palabras clave que aceptan las distintas propiedades CSS mencionadas en este apartado.

La recomendación de CSS 2.1 ofrece una gran cantidad de información adicional sobre las palabras clave de `font-size`*.

* http://www.w3.org/TR/CSS21/fonts.html#font-size-props

Demostración 1

De vez en cuando, este texto enlazará con un documento de demostración cada vez con más estilo que mostrará las propiedades CSS de las que hablamos en su uso real. El primero de los ejemplos muestra el documento HTML de muestra sin ningún tipo de estilo aplicado.

```
body { font-size: 14px; }
h1 { font-size: x-large; }
.sectionNote { font-size: medium; }
.attribution { font-size: small; }
```

Archivo fuente de: "Ejemplo sin estilos"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example1.html

Archivo fuente de: "Ejemplo con título redimensionado, atribuciones y parte central del texto aplicados"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example2.html

Archivo fuente de: "Ejemplo de hoja de estilo de la demostración 1"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_01.css

3.2.9. `font-family` y selección de tipos de letra

Cuando tengáis el texto con el tamaño que os guste, ya podéis pasar a seleccionar uno o dos tipos de letra. Esto se hace con la propiedad `font-family`, que se utiliza como se muestra a continuación en la demostración 2.

Cuando se ofrece un valor para la propiedad `font-family`, hay que seguir estas reglas:

1) Las tipografías se deben llamar exactamente como se llaman en la biblioteca de fuentes del ordenador cliente, utilizando como guía la fuente no variante.

2) Todas las tipografías con nombre deben separarse con comas, con espacio detrás o sin él.

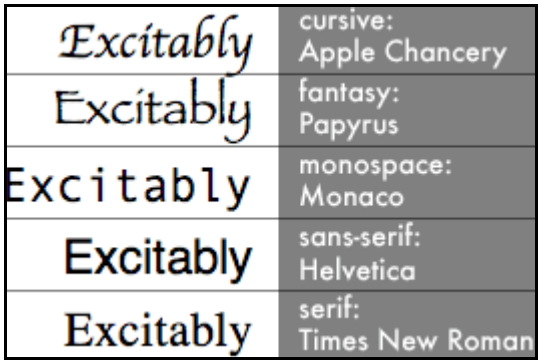
3) En los casos en que el nombre de una tipografía contenga más de una palabra, hay que poner comillas simples o dobles al principio y al final. **Ejemplo:** `'Times New Roman'`.

4) Hay que enumerar las tipografías por orden ascendente de posible disponibilidad. Por ejemplo, si queréis que los usuarios de Macintosh vean una página con el texto en Palatino, la declaración de valor de propiedad probablemente debería ser así: `font-family: Palatino, Georgia, serif;`. La Palatino es exacta-

mente la que queréis, pero puede ser que algunos usuarios no la tengan; la Georgia tiene más probabilidades de estar disponible y se parece a la Palatino; serif hace referencia a la fuente serif predeterminada genérica; si no se dispone de la Palatino ni de la Georgia, el sistema volverá a su fuente serif por defecto.

5) Como método de seguridad, el valor `font-family` siempre debería acabar con el nombre genérico apropiado, tal como se ha explicado anteriormente. En la figura 1 se muestran las tipografías que se utilizan en las familias genéricas en MacOS 10.5.

Figura 1



Las tipografías "genéricas" predeterminadas de MacOS 10.5 tal como quedan representadas a 24 píxeles en Safari 3.1.

En la recomendación del CSS 2.1 se enumeran varias tipografías más que pueden aplicarse a cada familia genérica*.

*** http://www.w3.org/TR/CSS21/fonts.html#generic-font-families**

Demostración 2

Ahora que el tamaño del texto es previsible, queremos optimizar las tipografías utilizadas. Lo mejor es aplicar una tipografía sans serif al título para que sea de fácil lectura y al texto en sí, una tipografía serif.

```
body { font-family: Palatino,'Palatino Linotype',Georgia,
            sans-serif; }
h1 { font-family: Helvetica,Arial,sans-serif; }
blockquote { font-family: Helvetica,Arial,sans-serif; }
```

Archivo fuente de: "Nuevas fuentes"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example3.html

Archivo fuente de: "Hoja de estilo de la demostración 2"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_02.css

3.2.10. `font-style`, `font-variant` y `font-weight`: cambiar los detalles

La propiedad `font-style` manipula la cursiva sin tener que utilizar el elemento `i`; sus tres valores válidos son `italic`, `oblique` y `normal`.

Los valores `italic` y `oblique` proporcionan resultados funcionalmente idénticos en las versiones más recientes de todos los navegadores web del mercado general, aunque hay una diferencia técnica significativa entre los dos estilos.

¿Por qué hay que utilizar la propiedad `font-style` cuando los elementos "em" e "i" ya parecen bastante adecuados?

Como ya hemos visto, cada uno de estos elementos tiene usos específicos. Además, la apariencia real del texto marcado con `em`, `i`, o cualquier otro elemento, puede variar según los diversos navegadores.

Hay situaciones en las que el uso de `em` y su hermano `strong` pueden requerir un punto de vista diferente. Por ejemplo, supongamos que queréis dar énfasis al texto haciéndolo más grande. Lo que habría que hacer para dar un mayor énfasis sería añadir cursiva, lo que daría lugar a reglas como las siguientes:

```
em {
  font-size: large;
  font-style: normal;
}
strong {
  font-size: large;
  font-weight: normal;
  font-style: italic;
}
```

Demostración 3

En el texto de demostración no hay palabras ni fragmentos en cursiva, y las atribuciones ya contienen la cursiva necesaria gracias al uso del elemento `cite`. Teniendo en cuenta la falta de opciones, el título es el mejor candidato para estar en cursiva.

```
h1 {font-style: italic; }
.sectionNote {font-style: normal; }
```

Archivo fuente de: "Ejemplo con título en cursiva"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example4.html

Archivo fuente de: "Hoja de estilo de la demostración 3"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_03.css

3.2.11. `font-variant` como herramienta para resaltar pasajes cortos

La propiedad `font-variant` tiene dos valores posibles, `small-caps` y `normal`. Algunos diseñadores utilizan versalitas o "small caps" para resaltar la frase inicial de un texto largo o para dar énfasis a la indicación de una cita, como por ejemplo:

ABANDONAD TODA ESPERANZA LOS QUE AQUÍ ENTRÁIS.

Demostración 4

```
opening {font-variant: small-caps; }
```

Archivo fuente de: "Ejemplo con la frase inicial en versalitas"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example5.html

Archivo fuente de: "Hoja de estilo de la demostración 4"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_04.css

3.2.12. `font-weight`: negrita y la falta de ésta

La propiedad `font-weight` (peso de la fuente) permite modificar el nivel de negrita de cualquier fragmento de texto al que se aplique.

Los valores habituales de la propiedad `font-weight` son `normal` y `bold`. La recomendación CSS proporciona los diversos valores* posibles.

*** https://www.w3.org/TR/CSS21/fonts.html#font-boldness**

Demostración 5

Poner el nombre de un autor en negrita es una técnica de diseño que se suele utilizar en las publicaciones periódicas, pero que aún encaja en varias situaciones de la web.

```
author {font-weight: bold; }
```

Archivo fuente de: "Ejemplo con el nombre del autor en negrita"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example6.html

Archivo fuente de: "Hoja de estilo de la demostración 5"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_05.css

3.2.13. La propiedad abreviada fuente

Muchas propiedades de texto pueden proporcionarse en el valor para una única propiedad, si procede, de una manera parecida a las propiedades de fondo.

A continuación, se muestra un ejemplo de regla abreviada de fuente:

```
h1 {font: italic normal bold x-large/1.167em Helvetica,Arial,sans-serif; }
```

Para obtener los mejores resultados, el valor dado a esta propiedad debería incluir los valores pretendidos para *todas* las propiedades individuales siguientes exactamente en este orden, separadas con espacios:

1) `font-style`
2) `font-variant`
3) `font-weight`
4) `font-size`, seguido, si es necesario, de una barra y del valor de `line-height` (podéis verlo a continuación).
5) `font-family`, que en este caso también puede ser una palabra reservada que especifica una fuente de sistema; los valores múltiples deberían estar separados por comas, pero no por espacios.

Tabla 3. Valores de palabra clave aceptados para las propiedades que se han tratado en este apartado

Propiedad	Valores
font-family	`cursive, fantasy, monospace, sans-serif, serif`
font-size	`xx-small, x-small, small, medium, large, x-large, xx-large`
font-style	`italic, normal, oblique`
font-variant	`normal, small-caps`
font-weight	`bold, normal`
line-height	`normal`
text-align	`center, justify, left, right`
text-decoration	`line-through, none, overline, underline`
text-transform	`capitalize, lowercase, none, uppercase`
white-space	`normal, nowrap, pre, pre-line, pre-wrap`

3.3. Las propiedades de texto y alineación de CSS. Modificar la composición

Un especialista en estilos que trabaje con letras y palabras trata con detalles: líneas, curvas, puntos, píxeles y otras partes *celulares* de un diseño. Sin embargo, un diseño es todo un conjunto; tiene componentes más grandes que se destacan mediante el control de la composición principalmente a través del

modelo de disposición de CSS. Además de este modelo de disposición, CSS también implementa propiedades de texto y alineación que afectan a la composición. El resto de este apartado trata sobre estas propiedades.

3.3.1. Alineación y justificación del texto

Como sucede con los entornos de procesamiento de texto, la propiedad `text-align` acepta cuatro valores de justificación: `left`, `right`, `center` y `justify`. Este último proporciona justificación completa: márgenes de texto visibles que son consistentes tanto por la izquierda como por la derecha de un bloque de texto.

Justificación adecuada del texto escrito en alfabetos occidentales

La mejor manera de utilizar las diferentes alineaciones disponibles es así:

- La **justificación izquierda** es más adecuada para fragmentos de texto largos.
- La **justificación derecha** es más adecuada para la columna de la izquierda del todo de tablas de datos y disposiciones de columna múltiple. Cuando la columna adyacente se deja justificada y colocada en el otro lado de un canal adecuadamente ancho, el resultado es mejorar la legibilidad de ambas columnas.
- La **justificación completa** funciona bien para bloques pequeños, como citas de bloques y textos de ejemplo. Cuando se utiliza en fragmentos largos como texto de ancho óptimo con márgenes anchos, la justificación completa también mejora la coherencia de la maquetación.
- El **centrado** se suele utilizar para los elementos de interfaz y listas en serie como las que se encuentran en los pies de página.

Demostración 6

Como la demostración se basa en un fragmento de texto de ficción extraído de un libro, ¿por qué no le aplicamos la justificación?

```
p { text-align: justify; }
blockquote p { text-align: left; }
```

Archivo fuente de: "Página con justificación completa del texto central del fragmento"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example7.html

Archivo fuente de: "Hoja de estilo de la demostración 6"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_06.css

3.3.2. Modificar el espaciado: las propiedades `letter-spacing` y `word-spacing`

El mismo nombre ya lo dice todo, es decir, estas propiedades permiten modificar la cantidad de espacio en blanco entre letras y palabras, respectivamente.

El uso más habitual de `letter-spacing` (espaciado de letra) es para dar un énfasis sutil parecido al efecto que proporciona `font-variant: small-caps;` también puede utilizarse para modificar sutilmente la composición de elementos de interfaz.

La propiedad `word-spacing` (espaciado de palabras) es mejor utilizarla para modificar deliberadamente la cantidad de palabras que es probable que aparezcan en una única línea de texto. Por ejemplo, puede utilizarse si tenéis un bloque de texto de la anchura habitual pero de tamaño de fuente atípica.

En la impresión, el espaciado de letras y el de palabras también se aplican *ad hoc* para evitar errores de composición, pero en la web esta técnica tiene una utilidad limitada.

Además de los valores de unidad, estas propiedades aceptan el valor `normal`.

Utilizar unidades em para un buen control

Cuando se cambia el espaciado entre letras de un texto, un pequeño espacio significa una gran distancia; el espaciado de letra predeterminado suele ser entre una décima y una vigésima parte de un em, de manera que los valores de `letter-spacing` que superen más del doble o la mitad del valor predeterminado pueden acabar haciendo que el texto sea ilegible.

Demostración 7

Al texto señal del que hablamos hacia el final le iría bien un poco de énfasis sutil... y, como la cita introductoria tiene un tamaño mayor, puede mejorarse mediante el espaciado entre palabras.

```
q { letter-spacing: .143em; }
.pullQuote { word-spacing: .143em; }
```

Archivo fuente de: "Página con espaciado entre letras en los saludos propuestos en el penúltimo párrafo del fragmento"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example8.html

Archivo fuente de: "Hoja de estilo de la demostración 7"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_07.css

3.3.3. Sangrar las primeras líneas: la propiedad `text-indent`

La propiedad `text-indent` (sangrado de texto) permite sangrar los párrafos de la misma manera que en el papel impreso. Además, esta propiedad y el hecho de que acepte valores negativos permite toda una serie de técnicas de maquetación avanzadas.

Los valores que acepta esta propiedad son los mismos que los que acepta `font-size`, además de `normal`.

Demostración 8

Por la misma razón que el fragmento tenía justificación completa, quizá debería tener sangrado en todas las primeras líneas de los párrafos.

```
p { text-indent: 1.429em; }
blockquote p { text-indent: 0; }
```

Archivo fuente de: "Página con sangrado inicial en los párrafos del texto central"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example9.html

Archivo fuente de: "Hoja de estilo de la demostración 8"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_08.css

3.3.4. Mayúsculas: La propiedad `text-transform`

De la misma manera que la propiedad `font-variant` proporciona acceso a las letras versalitas, la propiedad `text-transform` se ocupa específicamente de las mayúsculas. Los valores que permite cubren todas las mayúsculas, minúsculas o todo el texto inicial completamente en mayúsculas. Podéis ver la lista de los valores aceptados en la tabla 3.

Demostración 9

```
author {text-transform: uppercase; }
```

Archivo fuente de: "Página con el autor del fragmento destacado en versalitas"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example10.html

Archivo fuente de: "Hoja de estilo de la demostración 9"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_09.css

3.3.5. Aplicar estilos a los enlaces y mostrar las eliminaciones: la propiedad `text-decoration`

Esta propiedad permite poner líneas por encima, por debajo y a través del texto. Su uso más habitual es para manipular el subrayado por defecto de los enlaces, aunque en las wikis o en textos satíricos y en otros entornos también va muy bien para tachar palabras. En estos casos, se utilizan los elementos `ins` (insertar) y `del` (eliminar), que ofrecen estilos equivalentes a:

Incluso sin reglas de hoja de estilo hechas a medida, `ins` y `del` aplican estilo de etiquetado de la siguiente manera:

Benjamin Disraeli es famoso por sus dichos agudos, como "hay mentiras, grandes mentiras y estadística".

```
ins {
  text-decoration: underline;
}
del {
  text-decoration: line-through;
 }
```

Márgenes, no subrayados, con `acronym` y `abbr`

Algunos diseñadores suelen modificar el aspecto de los elementos `acronym` y `abbr` para que aparezcan con un subrayado que a primera vista parece una línea de puntos. No obstante, en realidad este efecto se consigue con un valor de `border-bottom`. (Recordad que algunos navegadores lo hacen automáticamente, pero otros, como IE 6, no.)

Demostración 10

```
.source a {text-decoration: none; }
```

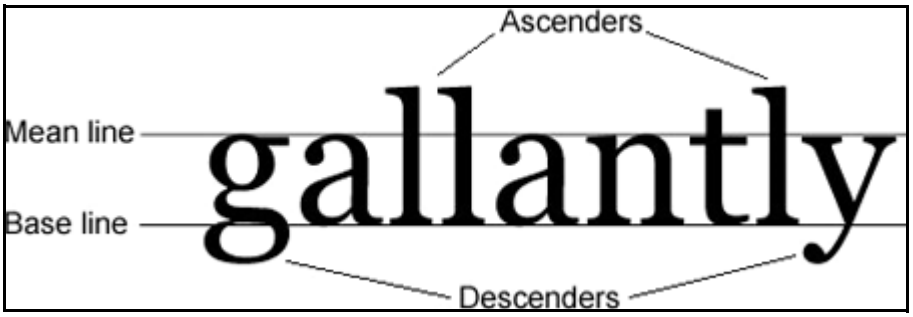
Archivo fuente de: "Página con eliminación del subrayado del enlace a fuente del texto de muestra"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example11.html

Archivo fuente de: "Hoja de estilo de la demostración 10"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_10.css

3.3.6. Ajuste del espacio interlineal y `line-height`

Es bien sabido que disponer de espacio en blanco entre líneas tiende a aumentar su facilidad de lectura porque el espacio adicional garantiza que las astas ascendentes y las astas descendentes (podéis ver la figura 2 para encontrar una explicación) en líneas adyacentes no competirán por la atención visual.

Figura 2



Las astas ascendentes son la parte de las letras que sobresale de la línea media del texto y las astas descendentes son la parte de las letras que se extiende por debajo de la línea base sobre la que descansa el texto.

Existe una tenue relación entre el `font-size` (el tamaño de la fuente) de un elemento y su `line-height` (interlineado), pero, por defecto, todos los agentes de usuario insertan algo de espacio interlineal en cada línea de texto, normalmente del 10 al 15% de la altura de las letras. Como esta cantidad por

defecto cambia de una tipografía a otra, la propiedad `line-height` (interlínea) acepta un valor de `normal` además de valores numéricos.

También cabe mencionar que, al contrario que la mayoría de las propiedades CSS, `line-height` (interlínea) acepta valores numéricos sin unidades, que después se representan como una proporción del valor básico predeterminado.

Demostración 11

Se ha hablado mucho de la relación entre espacio interlineal y legibilidad, de modo que aquí tenemos la demostración.

```
            p { line-height: 1.5; }
attribution { line-height: 1.5; }
```

Archivo fuente de: "Página con aplicación de una cantidad adecuada de interineado en un fragmento de texto"
http://mosaic.uoc.edu/ac/le/operafiles/29/copy_example12.html

Archivo fuente de: "Hoja de estilo de la demostración 1"
http://mosaic.uoc.edu/ac/le/operafiles/29/text_11.css

3.3.7. Suplantar `pre` y `br`: la propiedad `white-space`

Estrictamente hablando, los saltos de línea forzados son un elemento de presentación, aunque hay muchas circunstancias en las que son un elemento deseable para el diseño de un sitio. Junto con la costumbre de los navegadores de cortar las líneas en espacios arbitrarios, ejercer el nivel de control deseado únicamente con el etiquetado puede ser complicado.

El elemento `pre` sirve para hacer frente a estos retos, aunque presenta sus propios retos, razón por la que CSS ofrece la propiedad `white-space` (espacio en blanco). Los valores que acepta, que se enumeran en la tabla 3, permiten al diseñador elegir si el navegador debe representar los espacios en blanco y los saltos de línea que se han añadido al etiquetado de origen o que se han insertado como contenido generado.

La recomendación CSS 2.1 ofrece detalles exhaustivos sobre la implementación y el uso recomendados de la propiedad `white-space`* (espacio en blanco).

*** http://www.w3.org/TR/CSS21/text.html#white-space-prop**

Resumen

Un buen diseño de sitio requiere un nivel elevado de atención al detalle y el ajuste adecuado de la interacción de numerosos elementos, uno de los cuales es la tipografía.

La combinación de propiedades de fuente y texto que ofrecen las implementaciones de CSS de los navegadores actuales da prácticamente el nivel de control máximo sobre la tipografía que permite el hardware y depende del diseñador concienciado aprender a utilizarlas como es debido. Cuando estas propiedades se utilizan bien, se producen sitios que pueden aspirar a acercarse a los estándares de calidad de la tipografía que normalmente se asocian al medio impreso, que se ha ido desarrollando durante siglos, aunque la Web aún no hace ni una generación que existe.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. Decid tres elementos HTML **que no sean** `b` ni `i`, que por defecto se representen con fuentes variantes. Haced una descripción breve del propósito original de los elementos que habéis mencionado y explicad cómo el uso de una fuente variante es apropiado para los propósitos de estos elementos.

2. Comprobad de manera subjetiva la facilidad de lectura de un fragmento de un texto largo cualquiera con diferentes valores de `line-height` (interlínea). Resumid brevemente los resultados.

3. Aplicad `text-transform: uppercase;` a un único párrafo del fragmento utilizado en el ejercicio anterior y repetid la prueba subjetiva de legibilidad. Resumid brevemente los resultados.

4. Explicad brevemente las ventajas y desventajas del *antialiasing* utilizando el repaso de la tipografía de este apartado como ejemplo.

5. Describid el orden en el que deben especificarse las tipografías en un valor de `font-family`.

6. Sin consultar el material didáctico, elegid como mínimo tres propiedades de las descritas en este apartado y decid como mínimo un valor válido (aparte del predeterminado) de cada una. Demostrad o describid los resultados del uso de estas parejas de propiedades y valores en una hoja de estilo.

7. Cread un elemento, añadidle texto central y asignadle un valor de `font-size` (tamaño de la fuente) de `9px`. Abrid el documento que contiene este ele-

mento con Internet Explorer y pasad de un tamaño de fuente a otro de los que se proporcionan en el menú Visualiza > Tamaño del texto. Evaluad la conveniencia de estos resultados en sitios web con grandes cantidades de visitantes de mediana edad y de edad avanzada.

Lecturas complementarias

Bos, Bert y otros (2007). *Cascading style sheets level 2 revision 1 (CSS 2.1) specification*. World Wide Web Consortium. [Fecha de consulta: 28 de mayo del 2008].
http://www.w3.org/TR/2007/CR-CSS21-20070719

Chaparro, Barbara y otros (2004). *Reading online text: a comparison of four white space layouts*. Wichita State University Software Usablity Research Laboratory Usability News. [Fecha de consulta: 28 de mayo del 2008].
http://www.surl.org/usabilitynews/62/whitespace.asp

Dean, Paul (2008). "Extreme type terminology". I Love Typography, the Typography Blog. [Fecha de consulta: 28 de mayo del 2008].
http://ilovetypography.com/2008/03/21/extreme-type-terminology/

Horton, Sarah; Lynch, Patrick (2002). *Web style guide: basic principles for creating web sites* (2.ª edición). New Haven, Conn: Yale University Press.
http://www.webstyleguide.com/

Roselli, Adrian (2002). *A simple character entity chart*. Evolt. org. [Fecha de consulta: 28 de mayo del 2008].
http://www.evolt.org/article/A_Simple_Character_Entity_Chart/17/21234/

4. El modelo de composición de CSS: cajas, bordes, márgenes, relleno

Ben Henick

A primera vista, el modelo de composición de CSS es muy sencillo. Las cajas, los bordes y los márgenes son elementos bastante simples y la sintaxis de CSS ofrece una manera fácil de describir sus características.

Sin embargo, los motores de representación de los navegadores siguen una lista de reglas descritas en la Recomendación CSS 2.1 y unas cuantas propias. Por esta razón, hay muchos detalles que debemos comprender antes de poder añadir técnicas avanzadas al repertorio de un especialista en estilos.

En este apartado se presentan las propiedades de CSS que manipulan la composición de los elementos HTML como los bordes, los márgenes y muchos más. También se tratan algunas de las reglas mencionadas antes. En próximos apartados se hablará de la disposición avanzada de columnas y las técnicas de parrilla, y se tratará con más detalle la composición de los formularios, los elementos flotantes, la distancia y el posicionamiento.

> **Nota**
>
> A lo largo de todo el apartado encontraréis muchos ejemplos de código enlazados para demostrar las técnicas de las que se habla, pero si queréis trabajar con el código en vuestro ordenador, podéis descargaros todos los ejemplos de código.

Descargad los ejemplos en: "article30_examples.zip"
http://mosaic.uoc.edu/ac/le/operafiles/30/article30_examples.zip

4.1. Cambiar la composición: márgenes, bordes y relleno en CSS

Muchos elementos de HTML, como los elementos `div` y los títulos, se representan por defecto de manera que ocupen todo el ancho del lienzo del navegador y fuercen un salto de línea terminal, de modo que una serie de estos elementos se representaría como una pila de principio a fin sobre el lienzo del documento.

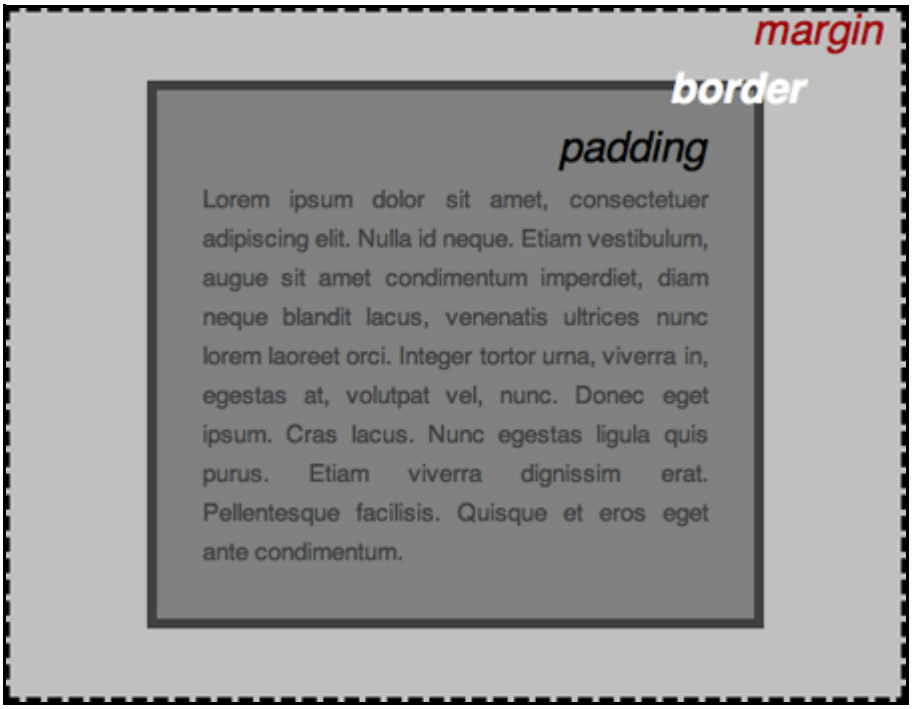
Sin embargo, los elementos de HTML y los estilos que el navegador suele asignarles no son adecuados para la gama completa de casos de uso que se pide a los desarrolladores que tengan en cuenta cuando hacen su trabajo. El modo como CSS y HTML se combinan se ha ajustado a una relación de "llenar los agujeros", de manera que las `class` e `id` puedan añadir significado semántico al etiquetado mientras que las reglas de hoja de estilo pueden cambiar la composición y la pre-

sentación del contenido de manera precisa, quizá incluso cancelando gran parte de los estilos predeterminados del navegador.

El control cuidado del espacio en blanco es una de las herramientas más importantes del diseñador y, según el autor de estas líneas, la más importante de todas. No obstante, el grado de control del espacio en blanco que aporta un valor de producción elevado al diseño de un sitio no está presente en las hojas de estilo predeterminadas de los navegadores, lo que significa que los especialistas en estilos suelen utilizar a menudo los márgenes, los bordes, el relleno y otras propiedades de CSS que se explican en este apartado.

Los márgenes (*margin*), los bordes (*border*) y el relleno (*padding*) se disponen tal como se muestra en la figura 1.

Figura 1



Representación explícita de las diferentes partes de una caja de elementos, con las respectivas etiquetas de las propiedades de CSS pertinentes

4.1.1. Colocar espacio en blanco en torno a un elemento: las propiedades `margin-top`, `margin-right`, `margin-bottom`, `margin-left` y `margin`

Los márgenes pueden especificarse de manera individual o en una regla abreviada. Además, la regla abreviada sigue permitiendo controlar cada uno de los bordes que rodean un objeto. Los valores válidos suelen especificarse en unidades `px` o `em` (píxeles o ems). En las hojas de estilo específicas de impresión pueden utilizarse las unidades `in`, `cm`, o `pt` (pulgadas, centímetros o puntos).

En todos los casos, `%` (porcentaje) es un valor válido, pero se debe utilizar con cuidado porque estos valores se calculan como una proporción de la anchura del elemento padre, y una previsión no suficientemente precisa de valores po-

Podéis ver la explicación del modelo de caja de CSS en el subapartado 3.2 de este módulo.

dría tener consecuencias no deseadas. Esta dificultad se explica con más detalle cuando se habla del modelo de caja de CSS.

Exceptuando las imágenes, los elementos en línea carecen de márgenes y no aceptan valores de margen.

En la tabla 2 que se muestra en el subapartado 4.1, hay una lista de los elementos en línea.

Márgenes automáticos

Dependiendo de las circunstancias, la provisión de un valor `auto` ordena al navegador que represente un margen según el valor proporcionado en su propia hoja de estilo. Sin embargo, cuando se aplica un margen así a un elemento con una anchura significativa, un margen `auto` provoca que todo el espacio disponible se represente como espacio en blanco.

Con la regla siguiente:

```
.narrowWaisted {
  width: 16.667em;
  margin: 1em auto 1em auto;
}
```

un elemento de bloque de la clase `narrowWaisted` (clase *cintura estrecha*) se centrará en medio del lienzo disponible.

También puede asignarse al margen derecho de un elemento aplicable un valor relativamente pequeño, mientras que al margen izquierdo se le asigna un valor `auto`.

Cuando se hace esto, un elemento así se representará casi pegado a la derecha.

Márgenes negativos

A todas las propiedades de márgenes se les pueden asignar valores negativos. En este caso, un margen adyacente puede ser "cancelado" a cualquier nivel. Si se aplica un valor lo suficientemente negativo a un elemento lo suficientemente grande, el elemento adyacente afectado puede acabar quedando por debajo del otro.

Como ejemplo, pongamos por caso los siguientes elementos sencillos `div`:

```
<div id="header"><h1>Lovely header</h1></div>
<div id="content"><p>Overlapping text is entirely unreadable</p></div>
```

Cuando se le aplican estilos con el CSS siguiente:

```
body {background-color:white font-family:Geneva, Arial, Helvetica, sans-serif;}
#header { background-color:yellow }
h1 { color:red; font-size:2em; }
```

se crea lo que se puede ver en la figura 2:

Figura 2

Los dos elementos de nuestro sencillo ejemplo. No tiene nada de especial.

Archivo fuente: "Negativemargins1 example"
http://mosaic.uoc.edu/ac/le/operafiles/30/negativemargins1.html

Pero ahora viene la parte interesante. Ahora añadiremos un margen negativo bastante grande encima del elemento inferior mediante la regla siguiente:

```
#content {margin-top:-3em;}
```

Esto provoca el efecto visual de cambiar el elemento de manera que se superponga con el título tal como se puede ver en la figura 3.

Figura 3

Cuando se le aplica un margen negativo, el elemento inferior pasa arriba y se superpone con el título.

Archivo fuente: "Negativemargins2 example"
http://mosaic.uoc.edu/ac/le/operafiles/30/negativemargins2.html

Colapso de márgenes

En los casos en los que dos elementos de bloque adyacentes y parecidos comparten márgenes que son mayores que cero, sólo se aplicará el margen mayor de los dos. Por ejemplo, pongamos por caso la regla siguiente:

```
p {
  margin: 1em auto 1.5em auto;
}
```

Si un documento que incluya esta regla de estilo se representa de manera literal, el margen final entre los dos párrafos seguidos sería de 2.5 em, es decir, la suma del margen inferior del párrafo 1 (1.5 em) y el margen superior del párrafo 2 (1 em). No obstante, a causa de la aplicación de márgenes colapsados, el margen entre ellos es sólo de 1.5 em.

Las listas y títulos son peculiares entre los elementos de bloque, de manera que sus márgenes no se colapsarán sobre los márgenes de los demás elementos de bloque.

Demostración 1

En el apartado anterior, de aplicación de estilos al texto, se hizo la tipografía de la sección introductoria de una historia de F. Scott Fitzgerald con la mayo-

ría de herramientas de CSS. De cara a la demostración de este apartado, se utiliza aquella misma página con algunos pequeños cambios (principalmente, la adición de un elemento contenedor en torno a todo el texto central). Los estilos del texto no cambian, pero se han eliminado los pocos estilos de composición que se habían aplicado a aquella demostración.

Para empezar, se añadirán márgenes a todos los elementos que los necesiten.

```
body { margin: 0; }
  #main { margin: 0 auto 0 auto; }
  h1 { margin: 0 0 1em 0; }
  .pullQuote { margin: auto 0 1em 1em; }
  p { margin: 0; }
  .attribution { margin: 0 0 1.5em 0; }
```

Archivo fuente de: "Ejemplo con sólo el mínimo de estilos"
http://mosaic.uoc.edu/ac/le/operafiles/30/demo_rev0.html

Archivo fuente de: "Hoja de estilo de inicio"
http://mosaic.uoc.edu/ac/le/operafiles/30/layout_00.css

Archivo fuente de: "Ejemplo con márgenes nuevos en `body`, título, texto destacado, contenedor de documentos y párrafos"
http://mosaic.uoc.edu/ac/le/operafiles/30/demo_rev1.html

Archivo fuente de: "Hoja de estilo de la demostración 1"
http://mosaic.uoc.edu/ac/le/operafiles/30/layout_01.css

4.1.2. Añadir un borde a un elemento: propiedades de borde

Hay una propiedad de `border` (borde) abreviada, pero sólo es útil cuando se quiere proporcionar un borde completo y consistente alrededor de los cuatro lados de un elemento. También es posible establecer el peso (anchura), estilo y color de cualquiera de los cuatro bordes de un elemento mediante cualquier combinación significativa de las propiedades siguientes:

- `border-width`
- `border-style`
- `border-color`
- `border-top`
- `border-top-width`
- `border-top-style`
- `border-top-color`
- `border-right`
- `border-right-width`
- `border-right-style`
- `border-right-color`
- `border-bottom`
- `border-bottom-width`
- `border-bottom-style`

- `border-bottom-color`
- `border-left`
- `border-left-width`
- `border-left-style`
- `border-left-color`

Las propiedades `border-width`

Estas propiedades se comportan exactamente como sería de esperar: asignan peso explícito a uno o más bordes.

La propiedad abreviada `border-width` (anchura de borde) acepta valores con la misma notación que la propiedad abreviada de `margin` (margen), pero no acepta valores porcentuales. Puede ser que en el futuro debáis redactar una regla como la siguiente:

```
td {
  border-width: 1px 0 0 1 px;
}
```

Las propiedades `border-style`

Figura 4

Los ocho estilos de borde más habituales

Las propiedades `border-style` (estilo de borde) suelen aceptar cualquiera de los valores siguientes:

- `dashed`. La longitud de las secciones de línea y la cantidad de espacio en blanco entre ellas las determina el navegador.
- `dotted`. La cantidad de espacio en blanco entre los puntos (que pueden tener cualquier forma con una proporción de 1) la determina el navegador.
- `double`. La anchura proporcionada se dividirá en tercios y se asignará en el orden lleno-negativo-lleno.
- `groove`. Se representará un `outset` inmediatamente dentro de y junto a un `inset`.
- `inset`. Al borde se le aplicará una sombra para que parezca que el elemento al que se aplica está incrustado en el lienzo.
- `none`. Equivalente a especificar una `-width` (anchura) de cero.
- `outset`. Al borde se le aplicará una sombra para que parezca que el elemento al que se aplica sobresale del lienzo.
- `ridge`. Se representará un `inset` inmediatamente dentro de y junto a un `outset`.
- `solid`. El borde es una línea continua sin sombra.

Cuando se utiliza la propiedad abreviada `border-style` (estilo de borde), ésta acepta hasta cuatro valores que se aplican de la misma manera que los valores abreviados `margin` (margen).

La práctica de oscurecer un borde (en lugar de omitirlo) se controla con las propiedades `-color`.

Las propiedades `border-color`

Para terminar, es posible establecer cualquier color en cualquier borde concreto, tanto con una única propiedad como las que se acaban de enumerar, como con la propiedad abreviada `border-color` (color de borde).

Podéis consultar la explicación de la propiedad abreviada de `margin` (margen) en el subapartado 1.4 para saber más detalles sobre los resultados de proporcionar menos de cuatro valores.

De la misma manera que `background-color`, `border-color` puede adoptar un valor de `transparent`. Esto puede ser útil para los casos límite que necesitan una composición consistente pero no un uso consistente de los bordes.

La propiedad abreviada `border` y sus cuatro primos, con más detalle

Al contrario que las distintas propiedades de borde `-width`, `-style` y `-color`, estas cinco propiedades permiten definir las tres características de los cuatro bordes de un objeto o de cualquier borde concreto en cualquier momento. Los valores abreviados de `border` (etc.) incluyen alguna o todas las propiedades de anchura, estilo y color que se aplican a un borde; la única limitación es que se debe hacer referencia o bien a un único lado cualquiera de un elemento o bien a los cuatro a la vez.

Tened en cuenta la siguiente regla de `border`:

```
#borderShorthandExample {
  border: 2px outset rgb(160,0,0);
  padding: .857em;
  background-color: rgb(255.224.224);
}
```

Un elemento al que se aplica la norma anterior tendrá exactamente el aspecto de este párrafo.

Cuando se omite un valor de una regla abreviada de `border`, el elemento representado mostrará un resultado por defecto:

- La **anchura del borde** la determinará el navegador.
- El **estilo del borde** será `solid`.
- El **color del borde** será idéntico al `color` aplicado al elemento en cuestión.

Crear reglas: la razón de ser de cinco propiedades abreviadas de borde en vez de una sola

Las "reglas" que se mencionan aquí son líneas trazadas sobre una composición y no directrices que haya que seguir. Estas líneas mejoran el contraste entre un elemento y el espacio que tiene alrededor y, en la mayoría de los casos, contribuyen a crear la ilusión de profundidad en una composición. Este último resultado se ejemplifica por la existencia de los estilos de borde `inset` y `outset`.

Si bien estos mismos efectos pueden conseguirse colocando bordes en torno a los cuatro lados de un elemento, la habilidad para trazar líneas definidas con precisión en una composición permite al diseñador alcanzar un control considerable sobre los detalles.

¿...Y por qué tantas propiedades? Sólo son bordes, ¿no?

Cuando se crea una composición que exige mucha pericia de un experto en estilos, siempre existe la necesidad de tener en cuenta casos límite.

Podéis ver los casos límite en el apartado 1 sobre los márgenes.

A causa del modo como se ejecutan los diseños de los sitios, os encontraréis muchos casos en los que algún elemento pueda llegar a tener propiedades estructu-

rales parecidas a otros elementos de un documento pero que tenga requisitos de presentación diferentes. En estos casos, tiene todo el sentido del mundo escribir una regla para el caso más común y reglas adicionales para cada uno de los casos límite. Por este motivo, existen los valores `auto` (automático) e `inherit` (heredar): para utilizar un estilo predeterminado como caso límite.

En el caso de los bordes, los casos límite puede ser que requieran la modificación de una sola característica de un borde de un único lado de un elemento, y cuando se es lo bastante sabio para seguir el principio KISS, normalmente lo mejor es cambiar únicamente los detalles que sean estrictamente necesarios.

Podéis ver el principio KISS en el subapartado 1.4 de este módulo.

Demostración 2

Determinadas secciones del documento deberían decorarse con reglas y bordes.

```
h1 { border-bottom: 1px solid rgb(153,153,153); }
.pullQuote { border: 1px solid rgb(153,153,153); }
```

Archivo fuente de: "Ejemplo con una regla inferior en el título y un borde alrededor del texto destacado"
http://mosaic.uoc.edu/ac/le/operafiles/30/demo_rev2.html

Archivo fuente de: "Hoja de estilo de la demostración 2"
http://mosaic.uoc.edu/ac/le/operafiles/30/layout_02.css

4.1.3. Cuando sólo con los márgenes no hay suficiente: propiedades `padding`

Encontraréis elementos con colores de fondo de tonos secundarios o destacados que necesitan canales entre el contenido y los márgenes. También puede ser que necesitéis incluir espacio entre los bordes y el texto que hay cerca.

En estos casos y en muchos otros os serán muy útiles las propiedades `padding` (relleno), `padding-top` (relleno superior), `padding-right` (relleno derecha), `padding-bottom` (relleno inferior) y `padding-left` (relleno izquierda). Estas propiedades insertan espacio negativo entre los márgenes o bordes de un elemento y su contenido. Podéis ver la figura 1 anterior como claro ejemplo de la relación entre márgenes, bordes y relleno.

Podéis ver la figura 1 en el subapartado 6.1 de este módulo.

Estas propiedades se comportan exactamente igual que las propiedades de margen pero con las excepciones siguientes:

Podéis ver las propiedades de margen en el subapartado 6.1 de este módulo.

- Los valores `auto` son funcionalmente inútiles en referencias a las propiedades de relleno.
- Los valores negativos de relleno son inválidos.
- El relleno nunca se colapsa.
- Los valores de margen no se aplican a los elementos en línea, pero los valores de relleno sí.

Demostración 3

Los canales deberían aplicarse a los elementos a los que ya se han aplicado bordes previamente.

```
body { padding: 0; }
  h1 { padding: .5em 0 .5em 0; }
  .pullQuote { padding: .5em; }
```

Archivo fuente de: "Ejemplo de inserción de canales adyacentes a los bordes previamente aplicados al título y al texto destacado"
http://mosaic.uoc.edu/ac/le/operafiles/30/demo_rev3.html

Archivo fuente de: "Hoja de estilo de la demostración 3"
http://mosaic.uoc.edu/ac/le/operafiles/30/layout_03.css

4.2. Trabajar con la anchura y altura de los elementos

Pueden modificarse las dimensiones de la mayoría de los elementos.

Anteriormente en el subapartado 6.1.1, cuando hablábamos de los márgenes `auto`, ya se ha demostrado que la mayoría de los elementos se pueden modificar.

Las propiedades CSS utilizadas para modificar las dimensiones de los elementos son: `width`, `height`, `min-width`, `max-width`, `min-height` y `max-height`. Estas propiedades pueden separarse de (o enlazarse a) las dimensiones del contenido de los elementos con la propiedad `overflow`.

También hay una propiedad `clip` (recorte) que esconde partes de un elemento dentro de los márgenes. Sin embargo, no hablaremos de ello en este apartado a causa de sus limitadas posibilidades de uso.

4.2.1. Conceptos básicos de `width` y `height`

En general, `width` y `height` producen exactamente los resultados que cabría esperar. No obstante, su uso implica algunas notas importantes.

Podéis ver una lista de los elementos que ignoran la aplicación de los valores `width` y `height` en el subapartado 6.4.1 de este módulo.

- **`width` y `height` no pueden aplicarse a elementos `inline`...** hay varios elementos (como `span`, `strong` y `em`) que ignoran la aplicación de los valores `width` y `height` en circunstancias normales.

- **... excepto a las imágenes, a las que se les puede asignar `width` y `height` aunque sean elementos en línea**. La Recomendación CSS 2.1 habla de las imágenes como elementos "sustituidos", lo que significa que los navegadores siempre deberían tratarlas como si tuvieran dimensiones estáticas. Por este motivo, estas dimensiones pueden modificarse arbitrariamente.

Podéis ver cómo las propiedades pueden afectar a las dimensiones funcionales de un elemento en el subapartado 6.3 de este módulo, que trata sobre el modelo de cajas de CSS.

- **`width` y `height` sólo son dos de las propiedades que pueden afectar a las dimensiones funcionales de un elemento**. Por lo tanto, es fácil encontrarse en situaciones en las que un elemento sea demasiado pequeño (normalmente demasiado estrecho) para contener el contenido como debería, lo que provoca errores.

- **Los errores de representación de Microsoft Internet Explorer (IE) hacen necesario especificar las parejas de valor/propiedad de `width` o `height` para algunos elementos.** Hay algunas peculiaridades del motor de representación de IE que sólo pueden resolverse con la fuerza bruta. La mayoría de estas peculiaridades son bien conocidas y se eliminarán de IE 8, pero hasta que esta versión sustituya a sus predecesoras en el mercado, este problema seguirá siendo un caso de prueba inevitable. PositionIsEverything.net* y la wiki CSS-Discuss** ofrecen mucha información sobre este asunto y técnicas para resolverlo.

* http://www.positioniseverything.net/
** http://css-discuss.incutio.com/

- **De vez en cuando, los algoritmos de redondeo provocan diferencias fuera de especificación en los navegadores que muestran los contenidos con medios de visualización LCD, LED o CRT (`type="screen"`).** El tipo de medio `screen` (pantalla) requiere que todas las unidades se conviertan en medidas de píxeles, de manera que la imagen puede quedar diferente en cada navegador.

4.2.2. `min-width`, `max-width`, `min-height` y `max-height`

En ocasiones os encontraréis situaciones en las que necesitáis limitar el tamaño de un elemento, generalmente para garantizar que una columna de tamaño proporcional conserve siempre una anchura legible. Las distintas propiedades `min-` y `max-` responden a este requisito. Del mismo modo que con `width` y `height`, los resultados que pueden esperarse de utilizar estas propiedades son bastante previsibles en realidad.

Demostración 4

Se colocaron márgenes `auto` a la izquierda y a la derecha del contenedor de la página. Ahora es necesaria una `width` para que los valores de margen tengan sentido. Además, el plan es asignar un valor `float` al texto destacado de manera que también obtenga una anchura.

```
#main {width: 42em; }
.pullQuote {width: 14em; }
```

Archivo fuente de: "Ejemplo de cambio de la anchura del contenedor del documento y del texto destacado"
http://mosaic.uoc.edu/ac/le/operafiles/30/demo_rev4.html

Archivo fuente de: "Hoja de estilo de la demostración 4"
http://mosaic.uoc.edu/ac/le/operafiles/30/layout_04.css

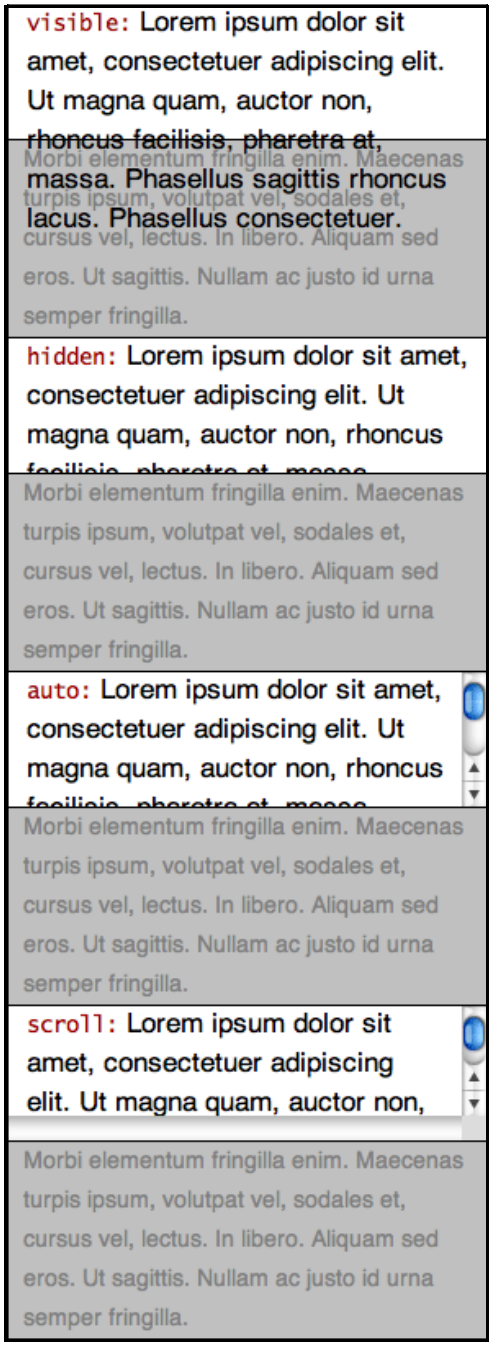
4.2.3. Desbordamiento: vallar el contenido o dejarlo libre

Cuando se establece la `width` o la `height` de un elemento, a veces es necesario pensar qué resultados son deseables en caso de que los contenidos de este elemento ocupen más espacio del que hay disponible. Esto es especialmente cier-

to para los sitios con contenido generado por el usuario y con especificaciones de composición estrictas.

La propiedad `overflow` (desbordamiento) y sus cuatro valores válidos (`visible`, `hidden`, `auto` y `scroll`) existen para hacer frente a estas circunstancias. En la figura 5 puede observarse el efecto que tienen cuando se aplican a un elemento cuyo contenido se desborda de los límites de su caja.

Figura 5



Los efectos de la propiedad de desbordamiento de CSS

Los resultados de los cuatro valores de `overflow`

- `visible` (**por defecto**). El contenido que sobrepasa las dimensiones de la caja de un elemento se muestra sin afectar al flujo o los márgenes de los

elementos adyacentes. En consecuencia, puede parecer que el contenido de un elemento choque con el contenido de un elemento adyacente. En subapartados posteriores, se habla de las técnicas para evitar esta situación y casos especiales provocados por problemas de representación en IE.

- `hidden`. Cualquier contenido que esté fuera de los límites de un elemento quedará escondido.

- `auto`. Las dimensiones de un elemento estarán limitadas igual que cuando se utiliza el valor `hidden` (escondido), pero se crearán las barras de desplazamiento que sean necesarias para que el contenido que no quepa sea accesible al visitante.

- `scroll`. Se incorporarán barras de desplazamiento verticales y horizontales al elemento aunque no sean necesarias.

4.3. Los modelos de caja de CSS: hacer que todo encaje

Ahora que ya hemos hablado de las propiedades fundamentales de composición, es el momento de descubrir cómo el navegador representa la anchura de un elemento según sus propiedades de CSS y cómo evitar que los elementos creen errores en las composiciones. Algunos resultados tendrán todo el sentido del mundo y otros parecerán horriblemente antiintuitivos. Para acabar de complicarlo todo, hay dos algoritmos de composición que se deben tener en cuenta: el modelo especificado por el World Wide Web Consortium (W3C) en la Recomendación de CSS 2.1 y el que se ha utilizado en las versiones antiguas de IE.

4.3.1. Elegir las unidades adecuadas para la composición

Tal como sucede con el texto, el tamaño de los elementos puede ajustarse con unidades proporcionales como `%` o `em`, o con unidades estáticas como `px`. Otro elemento que se debe tener en cuenta es que el lienzo del navegador siempre tiene un tamaño de un valor estático que no se puede asumir sin utilizar un script en el lado del cliente para recuperar el tamaño de la ventana o cambiarlo, técnicas que se ajustan mal a las exigencias de la accesibilidad, usabilidad y portabilidad de medios.

La regla principal de aplicar tamaño a los elementos: mezclar unidades proporcionales y estáticas con cuidado, o no hacerlo

El valor por defecto tanto de `width` como de `height` es `auto`, que es una directiva que significa: “Utiliza el espacio disponible”. El resultado de los elementos de bloque es que su `width` computada ocupa todo este espacio. En cuanto a la `height`, los elementos se amplían por defecto para incluir su contenido.

Si cambiáis los valores de `width` y `height`, debéis ir con mucho cuidado para garantizar que el contenido de un elemento encaje (con los márgenes, los bordes y el relleno) con la anchura que habéis especificado. La manera más fácil de hacerlo es llevar a cabo el proceso siguiente:

1) Pensad en el ancho máximo que pueda tener vuestra composición teniendo en cuenta las resoluciones de visualización habituales y/o el tamaño de la tipografía. En el momento de escribir estas líneas, esta anchura será normalmente de unos 800 o 1.024 píxeles. Cuanto más amplia sea la audiencia esperada de vuestro sitio, más probabilidades habrá de que se deba elegir el más pequeño de estos valores.

2) Cread un elemento contenedor para todo el documento ajustado a una anchura determinada inferior a la calculada en el paso #1.

3) Utilizad el mismo tipo de unidad cuando establezcáis las propiedades de composición de los elementos en el elemento contenedor creado en el paso #2.

Elegir el tipo de unidad adecuada para la composición: ventajas y desventajas

Tabla 1. Ventajas y desventajas del porcentaje, de em y de unidades de píxel que especifiquen propiedades de composición.

Unidad	Ventajas	Desventajas
em	Es más adecuado para crear cuadrículas de composición muy precisas en dos dimensiones. Cuando se utiliza en relación con contenedores de documentos, posibilita composiciones que se amplían o se contraen según el tamaño del texto central. Las dimensiones computadas de los elementos se convierten en fácilmente previsibles.	Puede ser que las unidades fraccionarias se amplíen o se contraigan con ligeras diferencias entre navegadores. Para conseguir los mejores resultados, todos los valores de font-size y line-height del documento deberían establecerse en valores explícitos y previsibles.
porcentaje	Es más adecuado para composiciones totalmente flexibles. Es más fácil para crear elementos como columnas iguales.	Para evitar errores, quizá se necesiten elementos contenedores adicionales. Puede resultar en elementos anchos de manera inaceptable o estrechos. Los resultados dependen mucho del contexto.
px	Ofrece el mayor control sobre la composición. Elimina la mayoría de la variación en la composición entre navegadores.	Es el menos adecuado para requisitos de accesibilidad y aceptación de todo tipo de medios. Es el más susceptible de causar errores.

4.3.2. Los componentes del modelo de cajas

En realidad, el modelo de cajas sólo es una serie de directivas que definen cómo interactúan entre sí las diferentes especificaciones de composición de un elemento. Los componentes que incluye el modelo de caja son:

1) Lienzo de documento.
2) Márgenes.
3) Bordes.

4) Relleno.
5) Anchuras y alturas de los elementos.
6) Propiedades de elementos hijo.

El último de estos elementos incluye a los otros cinco. Sin embargo, para simplificar, este subapartado se centrará en las relaciones sencillas de padre-hijo, y se reserva el tema de las interacciones de modelos de caja de múltiples niveles para futuros apartados que tratarán con más detalle los puntos de la composición de página.

4.3.3. El modelo de cajas del W3C: todo suma

La regla básica es que la anchura o altura computada de un elemento es igual a:

```
margin + border + padding + (width|height)
```

En muchos casos, la `width` y/o `height` se establecerán en su valor predeterminado de `auto`, lo que significa que la zona del lienzo reservada para el contenido es igual a:

```
lienzo_disponible – margin – padding – border
```

En una expresión de este tipo, `lienzo_disponible` es un valor discreto en sí (aunque a menudo autocomputado), del que se restan las cantidades correspondientes a márgenes, bordes y relleno. Este número es de gran importancia para la anchura de los elementos porque los errores de cálculo de anchura que cometa un diseñador tendrán la desagradable consecuencia de provocar que aparezca una barra de desplazamiento horizontal en la ventana del navegador. Adicionalmente, los navegadores siempre sitúan los elementos en el margen izquierdo del lienzo del navegador, ya que, en caso contrario, se desbordan fuera del margen derecho de la ventana del navegador si no se especifica lo contrario.

Fijaos en la siguiente regla de hoja de estilo:

```
#myLayoutColumn {
  width: 50em;
  margin: 1.5em auto 1.5em auto;
  border: .1em;
  padding: .9em;
}
```

Como ya hemos comentado en la explicación de las propiedades del margen, se puede esperar que `#myLayoutColumn` se centre en su elemento contenedor, independientemente de si el contenedor es `body` o algo creado por el grupo de producción.

Además, si la activación del "modo estricto" (mediante el uso de una declaración `!DOCTYPE` adecuada) causa que se utilice el modelo de cajas del W3C, también se puede esperar que la anchura *no marginal* computada prevista sea:

```
.1em + .9em + 50em + .9em + .1em = 52em
```

En los medios `screen`, el navegador tomará este valor, redondeará todos los valores de manera separada hasta el píxel más próximo y representará el resultado de acuerdo con ello.

Márgenes y relleno proporcionales en el modelo de cajas del W3C

Cuando se utiliza el modelo de cajas del W3C, los márgenes proporcionales y el relleno se computan en relación con la `width` computada del elemento contenedor. Por poner un ejemplo, si especificáis `margin: 20%` para un elemento incluido en un elemento que tiene 800 píxeles de anchura, el margen que se representa alrededor del primer elemento será de 160 píxeles por todos los lados (ya que el 20% de 800 son 160).

Si a este mismo elemento se le asigna `padding: 5%`, la anchura computada del contenido será de 400 píxeles:

```
20% + 5% + 5% + 20% = 50%
0.50 × 800 = 400
800 - 400 = 400
```

4.4. Trabajar con el flujo del documento

Los próximos apartados tratan de la creación de composiciones de múltiples columnas, de manera que debemos presentar tres propiedades de CSS que aún nos faltan en este apartado: `display`, `float` y `clear`.

4.4.1. Tipos de elementos y la propiedad `display`

Excepto las partes de `html`, `body` y `table`, todos los elementos de la Recomendación de HTML 4.01 que estén relacionados con contenido primario tienen

un tipo asociado “en línea” o “bloque”. Cada tipo determina el comportamiento de la composición por defecto de varias maneras.

a) En línea

- El texto y las imágenes que hay inmediatamente delante y/o detrás de elementos en línea se representan sobre una línea base común con el contenido del elemento en línea, a no ser que sean tan largos que se superpongan sobre el límite del elemento contenedor, en cuyo caso el contenido insertado acabará sobre una nueva línea base por debajo de la primera.

- Las líneas de texto dentro de elementos en línea se disponen con saltos de línea blandos según sea necesario (o según se pueda), excepto en los casos en los que este comportamiento se modifica por el uso de la propiedad `white-space` (espacio en blanco).

- Las propiedades `margin`, `width`, `height` y `float` en reglas de hoja de estilo aplicables a estos elementos (excepto `img` y `object`) se ignoran.

- Los elementos en línea sólo pueden contener texto u otros elementos en línea.

b) Bloque

- Estos elementos se representan como bloques discretos dentro de sus contenedores.

- A no ser que se les asigne un valor `float` de `left` o `right`, siempre se representarán con saltos de línea precedentes y posteriores.

- En general, los saltos de línea entre elementos de bloque anidados que no tienen ningún contenido entre ellos se colapsarán.

- Los elementos de bloque con una anchura de `auto` (la predeterminada) siempre se ampliarán para llenar toda la anchura que puedan.

La propiedad `display` (visualizar) tiene tres valores que se utilizan habitualmente: `block`, `inline` y `none`, de los cuales dos hacen referencia a los tipos de elemento correspondiente. El efecto de cambiar el valor de `display` de un elemento es provocar que un elemento en línea se comporte como un elemento de bloque, que un elemento de bloque se comporte como uno en línea o modificar la representación del documento como si el elemento (y todo su contenido) no existiera en absoluto.

De hecho, es fundamental saber qué elementos se corresponden con qué tipos por defecto. Las relaciones se exponen brevemente en la tabla 2:

Tabla 2. Elementos de HTML de uso frecuente y sus tipos. Sólo se colapsarán los márgenes entre dos elementos de bloque adyacentes del mismo subtipo.

Elemento	Tipo	Subtipo	Notas
`a`	en línea	especial	
`abbr`	en línea	frase	
`acronym`	en línea	frase	
`address`	bloque		En la práctica se comporta de manera parecida a `p`.
`blockquote`	bloque		Debe incluir como mínimo un elemento de bloque cuando el `!DOCTYPE` declarado es `Strict`.
`body`			Engloba todo el lienzo del documento; normalmente tiene un margen (en IE, Firefox y Safari) o un relleno (en Opera) de `10px` cuando el medio es `screen`.
`cite`	en línea	frase	
`div`	bloque		
`em`	en línea	frase	
`fieldset`	bloque		Normalmente se representa por defecto con `border: 1px black;`.
`form`	bloque		
`h1... h6`	bloque	título	
`input`	en línea	control de formulario	
`img`	en línea	especial	
`label`	en línea	control de formulario	
`li`	bloque		Tipo de elemento no especificado en la Definición de Tipo de Documento, pero este elemento puede contener elementos de bloque o insertados; la Recomendación CSS 2.1 completa reserva un valor `display` para elementos de lista.
`ol`	bloque	lista	
`p`	bloque		Sólo puede incluir elementos en línea, que normalmente se representan con márgenes superiores e inferiores.
`span`	en línea	especial	
`strong`	en línea	frase	
`table`	bloque		
`ul`	bloque	lista	

Demostración 5

¿Y si eliminamos la anotación "Prólogo" del título únicamente para esta demostración?

```
.sectionNote {display: none; }
```

Archivo fuente de: "Ejemplo con eliminación de la parte superflua de un apartado"
http://mosaic.uoc.edu/ac/le/operafiles/30/demo_rev5.html

Archivo fuente de: "Hoja de estilo de la demostración 5"
http://mosaic.uoc.edu/ac/le/operafiles/30/layout_05.css

4.4.2. Hacer que algunos elementos fluyan *en torno a* otros: la propiedad `float`

Hay una foto colocada a la izquierda de este párrafo. Prácticamente todos veréis que el texto siguiente fluye de manera natural a su alrededor, aunque puede que algunos primero debáis dejar de preguntaros por qué un famoso escritor de novelas de ciencia ficción decide pegar tocino a su gato con cinta adhesiva, por muy aburrido que esté. Los atributos del HTML pueden utilizarse para especificar el comportamiento de la composición que podéis observar, pero en este caso los resultados se han conseguido con CSS.

Tal como os podéis imaginar, la pareja propiedad/valor que lo hace posible es `float: left;`. En futuros apartados se tratarán las sutilezas de trabajar con elementos flotantes, pero, de momento, es necesario tocar este tema de pasada. `float: right` también es una pareja de propiedad/valor perfectamente útil, y para aquellas ocasiones en las que necesitáis contradecir una asignación de `class` que invoque `float`, podéis especificar `float: none`.

La propiedad `float` sí que tiene unas cuantas instrucciones de uso:

- Un valor `float` sólo tendrá relevancia si se aplica a un elemento de bloque con una `width` explícita.

- Las propiedades `float`, `clear` y `margin` aparecen *juntas* en las reglas de hojas de estilo para crear columnas en una composición.

- Es complicado hacer que un elemento flotante se estire hasta llegar a la parte inferior de su contenedor, pero no es imposible. La manera habitual de hacerlo se denomina *faux-columns**.

*** http://www.alistapart.com/articles/fauxcolumns/**

Demostración 6

Ya hemos hablado de la colocación de un valor `float` en el texto destacado, de manera que ahora lo haremos y veremos cómo queda. Y, ya que estamos, añadámosle un poco de color de fondo para que se distinga mejor del contenido principal.

```
.pullQuote { float: right;
background-color: rgb(204,204,204); }
```

Archivo fuente de: "Ejemplo de hacer 'flotar' el texto destacado hacia el margen derecho"
http://mosaic.uoc.edu/ac/le/operafiles/30/demo_rev6.html

Archivo fuente de: "Hoja de estilo de la demostración 6"
http://mosaic.uoc.edu/ac/le/operafiles/30/layout_06.css

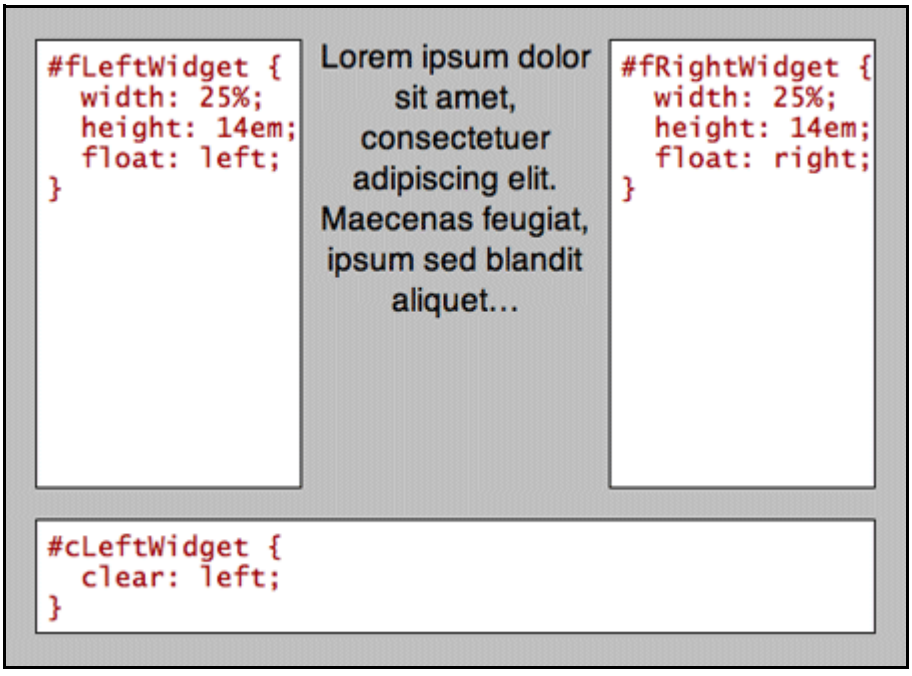
4.4.3. Forzar elementos *por debajo* de sus predecesores flotantes: la propiedad `clear`

De la misma manera que la propiedad `float`, a la propiedad `clear` puede asignársele uno de los valores `left`, `right` o `none`. El valor `both` (ambos) también es posible.

Aunque la propiedad `float` rige cómo debería fluir el contenido de los elementos subsiguientes a su alrededor, la propiedad `clear` rige cómo debe fluir un elemento en torno a todos sus vecinos; en la mayoría de los casos, no debería fluir en absoluto.

La figura 6 demuestra el funcionamiento de `clear: left;` en una composición en la que a dos elementos consecutivos se les han asignado valores idénticos de `height` y valores de `float` de `left` y `right`:

Figura 6

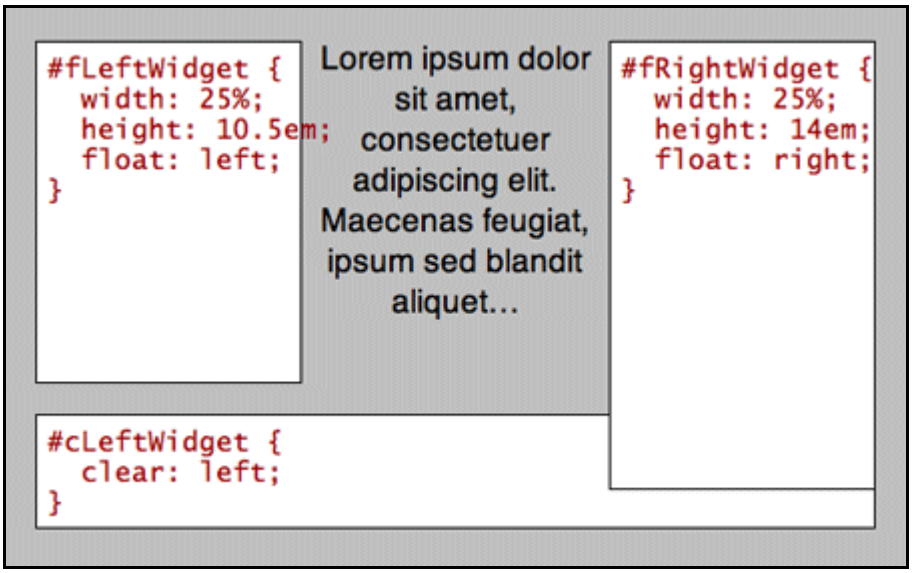


`clear:left` permite que la caja inferior corte las dos columnas porque son de la misma altura.

En la demostración anterior, el flujo por defecto de `#cLeftWidget` lo colocaría justo por debajo del texto en latín, es decir, entre `#fLeftWidget` y `#fRightWidget`.

Pensad qué sucede cuando el primero de la misma colección de elementos se hace más corto que su hermano que toca a la derecha, tal como se ve en la figura 7.

Figura 7



Cuando la columna derecha es más larga que la columna izquierda, `clear:left` no cortará las dos columnas y, por lo tanto, hay que utilizar `clear:both`.

En el primer ejemplo, el valor `clear` del elemento final se establece en `left` para demostrar este hecho: como los dos elementos con `float` son igual de altos, el elemento con `clear` se coloca automáticamente por debajo de los dos. Sin embargo, el segundo ejemplo demuestra que para conseguir el mismo resultado con elementos con `float` de alturas diferentes, hay que utilizar `clear: both;`.

Esta información sobre la propiedad `clear` está pensada sólo como una pequeña introducción a sus efectos, ya que los detalles de la técnica asociada a su uso se tratarán en apartados posteriores.

Resumen

Entre las diferencias entre motores de representación, la necesidad de incluir una amplia gama de terreno definido tradicionalmente y la inhabilidad para predecir las dimensiones exactas de una ventana de navegador, la composición de documentos web está llena de dificultades y riesgos. A pesar de todo, el nivel habitual de soporte de CSS ha avanzado hasta el punto de que no es difícil conseguir que los documentos web den resultados aceptables en cualquier navegador.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿En qué circunstancias es mejor utilizar el valor abreviado como `margin` o una única propiedad de margen como `margin-top`?

2. Cuando las propiedades abreviadas de `margin`, `padding` y `border-width` se presentan con los cuatro valores, ¿en qué orden se aplican estos valores a los cuatro lados de un elemento?

3. Si quisierais colocar una regla bajo el texto de cada título de un documento, ¿qué propiedad utilizaríais?

4. ¿Qué valor de `border-style` utilizaríais para hacer que un elemento parezca un botón de interfaz?

5. *Sí o no:* ¿especificar un borde alrededor de un elemento también creará automáticamente un canal en torno al elemento?

6. Si creáis un elemento que no sea tan ancho como su contenedor, ¿qué pareja de propiedad/valor debéis establecer para garantizar que el elemento quede centrado horizontalmente en su contenedor?

7. *Sí o no:* si colocáis un elemento contenedor dentro de `body` y establecéis la `width` en un valor superior a `100%`, ¿cambiará el comportamiento del lienzo del documento?

8. Si una imagen es demasiado grande para el elemento contenedor, ¿qué pareja de propiedad/valor utilizaríais para garantizar que no se creen errores en la composición de la página y por qué?

9. Si asignáis un valor de `display` de `block` a un elemento `a` (enlace) y dais una altura y una anchura razonables a este elemento, ¿cómo cambia el comportamiento del enlace cuando se pasa el ratón por encima en un medio de visualización `screen`?

10. En circunstancias normales, un elemento de bloque se amplía hasta llenar la anchura de su contenedor (menos márgenes, bordes y relleno). Por defecto, ¿cambia realmente este comportamiento cuando este elemento va precedido de un elemento con `float`, o sólo parece que cambia?

11. Si pretendéis aplicar un valor `float` a un elemento, ¿qué otra propiedad también debéis establecer en este elemento?

12. Si quisierais estar del todo seguros de que un elemento siempre se ampliará para llenar la anchura de su contenedor, ¿qué parejas de propiedad/valor estableceríais?

Lecturas complementarias

Bergevin, Holly; Gallant, John (2006). *Explorer exposed. Position Is Everything*. [Fecha de consulta: 1 de julio del 2008].
http://positioniseverything.net/explorer.html

Bos, Bert y otros (2007). *Cascading style sheets level 2 revision 1 (CSS 2.1) specification*. World Wide Web Consortium. [Fecha de consulta: 30 de junio del 2008].
http://www.w3.org/TR/2007/CR-CSS21-20070719

Raggett, Dave y otros (1999). *HTML 4.01 specification*. World Wide Web Consortium. [Fecha de consulta: 30 de junio del 2008].
http://www.w3.org/TR/1999/REC-html401-19991224

Raymond, Eric; Steele, Guy (eds.) (2003). *Brute force. The Jargon File (Version 4.4.7)*. [Fecha de consulta: 30 de junio del 2008].
http://www.catb.org/jargon/html/B/brute-force.html

Scalzi, John (2006). *Clearly you people thought I was kidding. Whatever.* [Fecha de consulta: 30 de junio del 2008].
http://www.scalzi.com/whatever/004457.html

5. Imágenes de fondo en CSS

Nicole Sullivan

¡Admitidlo! Desde el primer apartado de este curso os habéis estado muriendo de ganas de aprender cómo conseguir que vuestra web quede impresionante y fabulosa. ¡Incluso es posible que os hayáis saltado los otros apartados para llegar a éste directamente!

Las imágenes de fondo sirven para hacer que vuestro sitio quede atractivo, pero quizá os sorprenda saber que están muy relacionadas con los conceptos fundamentales que ya habéis aprendido.

Tal como ya habéis visto en esta asignatura, uno de los cambios más importantes que aporta el CSS es la capacidad para separar la **presentación**, o el aspecto de las cosas, de la **semántica**, o el significado de las cosas. La imagen de fondo de CSS es una de las herramientas más importantes que podéis utilizar porque permite aplicar imágenes decorativas en determinadas partes de vuestro HTML sin añadir peso adicional al HTML. Antes, los creadores web (¡como vosotros!) se veían obligados a llenar el código de etiquetas `img`.

El CSS, y sobre todo la propiedad `background` (fondo), hacen que vuestro HTML deje de estar abarrotado de elementos presentacionales. De esta manera, rediseñar un sitio o realizar otras transiciones de la vida de una página web creada con métodos modernos puede hacerse de una manera mucho más sencilla. Podréis actualizar toda la página cambiando únicamente la hoja de estilo en lugar de tener que grabar todas y cada una de las páginas HTML. Dependiendo del tamaño del sitio, esto puede ahorraros una gran cantidad de tiempo.

En este apartado veremos los conceptos básicos del funcionamiento de las imágenes de fondo de CSS, incluida la aplicación de una imagen de fondo a través de CSS, cómo ajustar su colocación, repetirla en vertical u horizontal y combinar imágenes de fondo con *sprites* CSS* para mejorar el rendimiento del sitio**.

*** http://www.alistapart.com/articles/sprites/**

**** http://developer.yahoo.com/performance/index.html**

5.1. ¿Cómo funciona?

El CSS para fondo se divide en varias propiedades. Utilizando estas propiedades, como `position` y `color`, podéis empezar a controlar el aspecto de vuestra página. En este apartado, repasaremos detalladamente las imágenes de fondo de CSS y crearemos un mensaje de alerta como ejemplo paso a paso.

Antes de nada, debemos saber más cosas sobre las diferentes propiedades que podemos utilizar.

5.1.1. Propiedades de fondo

Tabla 1. Propiedades de fondo

Propiedad	Definición	Descripción
`background-color`	Establece el color de fondo del elemento de HTML.	Hay varias maneras de indicar el `background-color` o color de fondo, como incluir valores y palabras clave RGB. La mayoría de la gente utiliza la notación hexadecimal, un símbolo de almohadilla o barra (#) seguido de seis caracteres. La primera pareja indica los niveles rojos y la segunda y la tercera indican los niveles verde y azul respectivamente: #RRGGBB. Hay muchas herramientas para elegir colores que pueden ayudaros a encontrar la notación hexadecimal de un color determinado. El rojo puro, por ejemplo, sería el #FF0000.  Los valores válidos incluyen un valor de `color`, `transparent` o `inherit`.
`image`	Indica la ruta o URL de la imagen de fondo.	Establece la `background-image` o imagen de fondo mostrando al navegador dónde encontrar la imagen utilizando la URL. Por ejemplo: `url(alert.png)`. Fijaos en que la ruta va precedida de la palabra clave `url` y entre paréntesis. Esta sintaxis es importante para que el navegador entienda que queréis indicar una ubicación. Los valores válidos incluyen una `URL`, `none` o `inherit`.
`repeat`	Indica en qué dirección se debe repetir la imagen de fondo.	Las imágenes pueden repetirse en vertical u horizontal, o en ambas direcciones, para llenar toda la anchura o altura de un elemento HTML. Utilizad `background-repeat` para indicar al navegador que repita una imagen de fondo. Los valores válidos son `repeat`, `repeat-x`, `repeat-y` y `no-repeat`.
`attachment`	Define el comportamiento de la imagen de fondo cuando el usuario se desplaza.	Las imágenes pueden desplazarse con el contenido o quedarse fijas en la pantalla de visualización. Los valores válidos son: `scroll`, `fixed` e `inherit`.
`position`	Indica al navegador dónde colocar la imagen de fondo.	Las imágenes pueden mostrarse en cualquier punto dentro de los bordes del elemento de HTML en el que estén aplicadas. Utilizad `background-position` para colocar las imágenes con precisión o para crear efectos visuales y capas. Hay muchas maneras útiles de indicar la posición, las palabras clave y los valores numéricos del fondo. Las palabras clave (como `top` y `bottom`) son muy útiles y fáciles de leer. Los valores en píxeles son muy precisos pero no se adaptan a los cambios de altura y anchura. Los valores de píxeles negativos son muy útiles cuando se utilizan *sprites* CSS, como veremos más adelante. Cuando se utilizan *sprites* y píxeles, el punto inicial siempre es el extremo superior izquierdo del elemento HTML, aunque la manera de funcionar de la colocación de la imagen es algo diferente con píxeles o con porcentajes. Los píxeles siempre mueven la imagen un número concreto de píxeles hacia la parte inferior derecha de la caja contenedora (o hacia la parte superior izquierda si son valores negativos), sea cual sea el tamaño de la imagen y de la caja contenedora. Los porcentajes, en cambio, mueven la imagen un porcentaje de la diferencia entre la caja contenedora y el tamaño de la imagen. Si la imagen y la caja contenedora son igual de grandes, los porcentajes no moverán la imagen en absoluto. Los valores válidos incluyen `length` (normalmente en píxeles), `percentage` (de la anchura del elemento) y las palabras clave `top`, `right`, `bottom`, `left` y `center`. Fijaos en que `center` se puede utilizar para indicar un centro vertical u horizontal. Fijaos también en que podéis mezclar porcentajes y píxeles en las reglas, pero no palabras clave y píxeles ni palabras clave y porcentajes.

Propiedad	Definición	Descripción

`background`	La propiedad abreviada que se puede utilizar para describir todo el resto de propiedades de una línea.	Las propiedades abreviadas son realmente prácticas. La mayoría de los desarrolladores las utilizan para mantener el CSS lo más sencillo posible y para agrupar propiedades relacionadas. Se puede escribir una regla general utilizando la abreviación y después anularla según sea necesario con propiedades específicas.
		Las propiedades siempre se deberían indicar en el mismo orden para que los navegadores puedan interpretar fácilmente los estilos.
		`1.color`
		`2.url`
		`3.repeat`
		`4.attachment` (no se utiliza casi nunca, se puede omitir)
		`5.horizontal-position`
		`6.vertical-position`
		Un ejemplo de esta abreviación con todas las propiedades utilizadas (excepto `attachment`):
		`background: green url(logo.gif) no repeat left top;`

5.2. Crear un mensaje de alerta

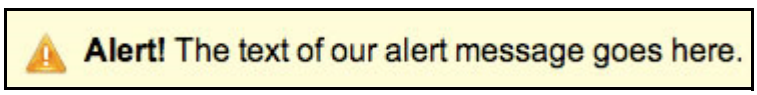
Ahora que ya hemos visto la sintaxis básica necesaria, os enseñaremos cómo crear un ejemplo completo de caja de alerta que servirá para demostrar todos los aspectos de las imágenes de fondo.

5.2.1. El diseño

Pongamos por caso que un diseñador gráfico os ha presentado un boceto visual del mensaje de alerta que queréis crear para vuestro sitio web. Fijaos en que la alerta tiene el fondo de color naranja claro para diferenciarlo de los párrafos que lo rodean. También tiene un icono de alerta a diez píxeles del extremo superior izquierdo.

Fijaos en que el boceto sólo tiene una línea de texto, pero que en otros casos puede tener más. Uno de los rasgos más importantes del desarrollador web es prever cómo acabará evolucionando un diseño. En parte, respetar la visión artística de un sitio supone pensar en la consistencia desde que se inicia hasta que se rediseña. Por lo tanto, el mensaje de alerta podría incluir más de una línea de texto o incluso múltiples párrafos, listas u otros elementos HTML. Deberíais intentar ser tan agnósticos en cuestión de elementos como os sea posible. De esta manera, aumentarán las probabilidades de reutilización del código y el sitio web tendrá la máxima velocidad y eficiencia. El boceto es el que se ve en la figura 1:

Figura 1



El boceto del diseñador gráfico de la caja de alerta

El diseñador también os ha proporcionado el icono que debéis utilizar, como se muestra en la figura 2:

Figura 2

El icono de alerta

5.2.2. El código

Basándoos en todo lo que habéis aprendido de los fondos de CSS en la primera parte de este apartado, ya podéis ir pensando en cómo crear este mensaje de aviso. Os animamos a que lo probéis ahora mismo y que después comparéis el resultado con nuestro ejemplo.

Muy bien, ¿ya lo habéis probado? Vayamos paso a paso. Haced pruebas con el código, aumentad o reducid los valores e intentad probar alternativas. También puede ser que queráis ir siguiendo todos los pasos escribiendo cada línea nueva de código en una herramienta como por ejemplo Opera Dragonfly o Firebug para poder ir viendo los resultados de cada paso inmediatamente.

> **Nota**
>
> En la versión web cada copia de pantalla tiene un enlace a ejemplos de código para que podáis comprobar la fuente en cada paso.

Crear el vínculo o selector de CSS

Para empezar, debéis crear una clase `alert` para que el CSS se pueda enlazar con él. Cread los ficheros esqueleto nuevos de CSS y HTML, enlazad el CSS en el fichero HTML y añadidle el código siguiente:

El CSS:

```
.alert {... }
```

El HTML:

```
<p class="alert">
  <strong>Alert!</strong> The text of our alert message goes here.
</p>
```

En este caso, hemos aplicado estilos a la alerta con un `class` y no con un `id` porque podría tener *más de una alerta* en la página, por ejemplo, en un elemento de formulario con varios errores. El CSS se debe hacer lo más flexible que se pueda y limitarlo todo a fin de que se corresponda al diseño cuando se crea el HTML.

Muy bien, pues ya tenéis unas bases, pero todavía parece un párrafo normal porque todavía no le habéis añadido ningún estilo. Hagámoslo.

> **Nota**
>
> Hemos decidido intencionadamente no limitar la clase `alert` a párrafos; las cajas de alerta también podrían contener fácilmente otros elementos. Deberíais dejar que el CSS tenga la máxima flexibilidad posible.

Añadir el color de fondo

Ya habéis aprendido cómo utilizar el color de fondo con los estilos de texto. Los mismos principios se aplican a cualquier elemento HTML y pueden combinarse con imágenes de fondo para crear efectos visuales. Si el color de fondo no se ha establecido ni heredado, será transparente por defecto.

Podéis ver cómo utilizar el color de fondo con los estilos de texto en el apartado 3 de este módulo.

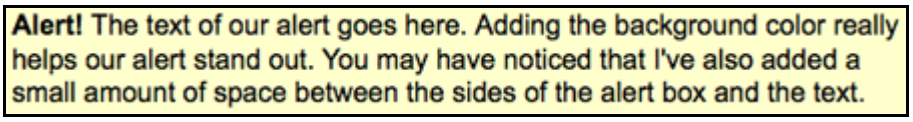
Añadimos el color de fondo naranja claro a la caja de alerta para que resalte entre el texto que tiene alrededor. No debería ser demasiado oscuro porque es importante que mantengáis un nivel razonable de contraste entre el texto y el color de fondo*. Añadid la propiedad siguiente a vuestra regla de CSS:

* http://www.w3.org/TR/2006/WD-WCAG20-20060427/guidelines.html#visual-audio-contrast

```
.alert{background-color: #FFFFCC;}
```

Ahora la caja de alerta se parecerá más a la figura 3:

Figura 3



Una caja de alerta con color de fondo añadido

Archivo fuente de: "Figura 3"
http://mosaic.uoc.edu/ac/le/operafiles/31/2_color.html

Aplicar la imagen de fondo

Pasemos a añadir la imagen. La ruta a la imagen de fondo debe estar entre `url()`, como se observa en el código siguiente. Añadid la línea resaltada a la regla de CSS:

```
.alert{
  background-color: #FFFFCC;
  background-image: url(alert.png);
}
```

Ahora la caja de alerta quedará como en la figura 4:

Figura 4

Se ha añadido la imagen de fondo, pero la repetición queda horrible.

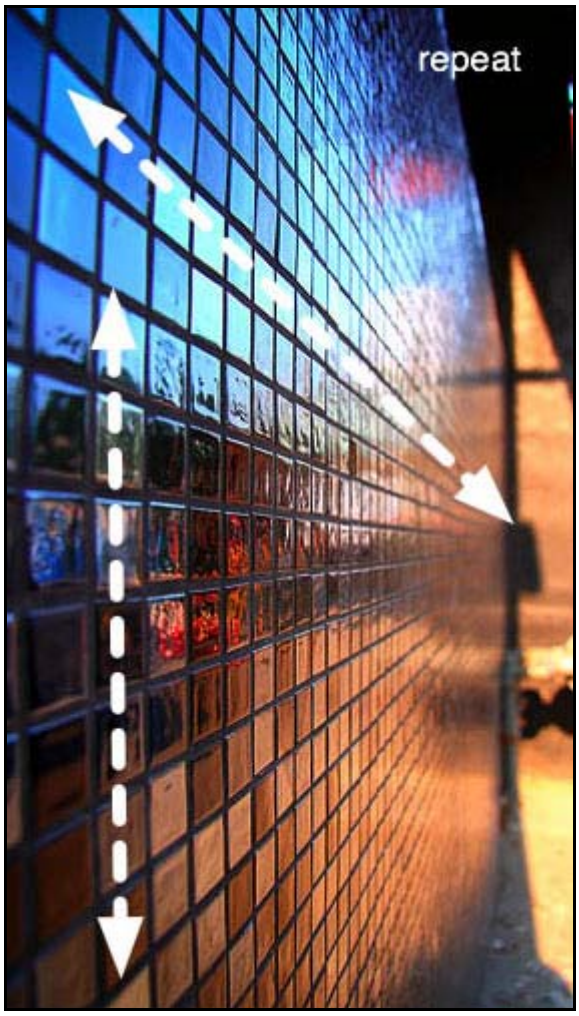
Archivo fuente de: "Figura 4"
http://mosaic.uoc.edu/ac/le/operafiles/31/3_image.html

Recordad que cada propiedad de fondo tiene un valor por defecto que se le aplicará si no especificáis ningún valor. Seguramente ya os habréis dado cuenta de que la imagen se repite por toda la alerta como si fueran las teselas de un mosaico. ¿Cuál es la conclusión de todo esto? Las imágenes de fondo están configuradas por defecto para repetirse tanto horizontal como verticalmente.

Los fondos que se repiten son especialmente útiles para gradaciones de colores y patrones que llenan la pantalla o un elemento de HTML en concreto, pero en este caso no es lo que queremos.

Controlar la repetición del fondo

Figura 5



De manera parecida a nuestra imagen de fondo, estas teselas se repiten tanto en horizontal como en vertical.

Leer especificaciones puede ser una tarea intimidante, pero la especificación es muy buen lugar para averiguar cómo se supone que funciona el CSS antes de pasar ver con detalle la miríada de diferencias existentes entre navegadores. Echadle un vistazo a la parte de los colores y los fondos de la especificación del W3C* e intentad encontrar la palabra clave para utilizarlos cuando no queráis que se repita una imagen de fondo. A continuación, la utilizaremos en un ejemplo.

*** http://www.w3.org/TR/CSS21/colors.html**

¿Ya la habéis encontrado? Fijaos en que existe una sección para cada propiedad de fondo, `background-repeat*` incluida. En **Value** veréis todas las opciones posibles, incluyendo: `repeat`, `repeat-x`, `repeat-y`, `no-repeat` e `inherit`. Por defecto, las imágenes de fondo (**iniciales**) se repiten. No se especifica ninguna dirección, de modo que la imagen se repetirá tanto ho-

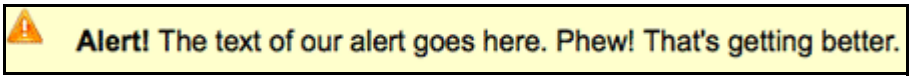
*** http://www.w3.org/TR/CSS21/colors.html#propdef-background-repeat**

rizontal como verticalmente. Seguramente ya os habréis imaginado que `no-repeat` (sin repetición) es el valor que necesitáis para impedir que la imagen se repita en cualquier dirección. Añadid la siguiente línea resaltada a la regla de CSS:

```
.alert{
  background-color: #FFFFCC;
  background-image: url(alert.png);
  background-repeat: no-repeat;
}
```

Ahora la caja de alerta quedará como en la figura 6:

Figura 6



La caja de alerta con una única copia de la imagen de fondo (sin repeticiones)

Archivo fuente de: "Figura 6"
http://mosaic.uoc.edu/ac/le/operafiles/31/4_repeat.html

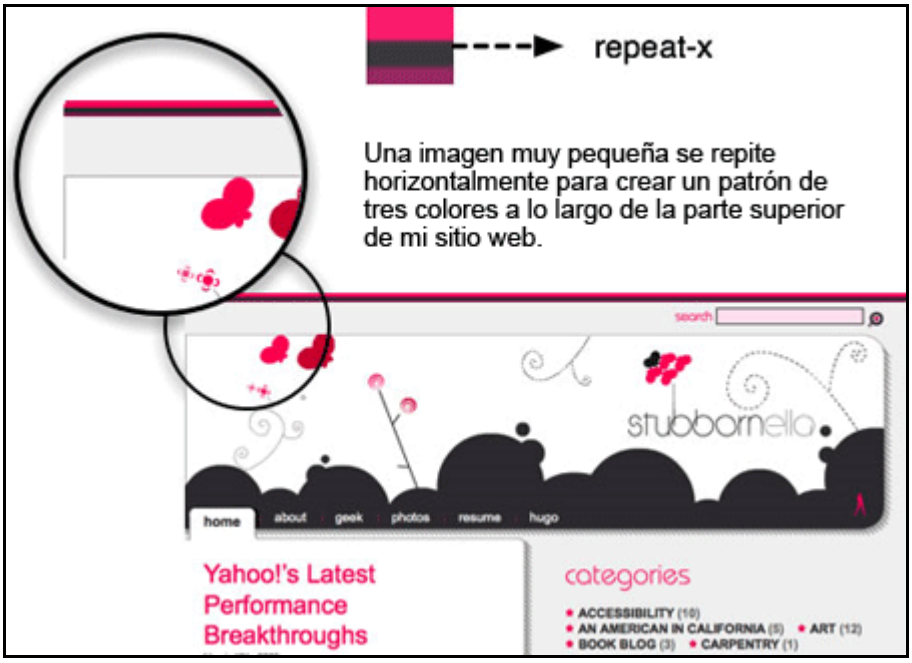
Además, también podéis repetir en ambas direcciones (como si fueran teselas de un mosaico) o en ninguna. Las gradaciones suelen repetirse horizontal o verticalmente (podéis ver la figura 7). No es necesario que sepáis el tamaño del elemento HTML; sólo debéis cortar un trozo de la gradación y hacer que se repita en la dirección que queráis; ya sea "x" para horizontal o "y" para vertical. Los patrones a menudo se repiten en ambas direcciones y los iconos normalmente no se repiten. Más adelante examinaréis `background-repeat` con más detenimiento en otro ejemplo.

Figura 7

Las teselas o baldosas de este ejemplo sólo se repiten en horizontal.

Fijaos ahora en un ejemplo práctico de mi web, observad la figura 8:

Figura 8



Un ejemplo de imagen repetida de mi web*

Archivo fuente de: "Figura 8"
http://mosaic.uoc.edu/ac/le/operafiles/31/5_attachment.html

*** http://mosaic.uoc.edu/ac/le/operafiles/31/5_attachment.html**

El CSS que utilizamos para añadir este efecto decorativo es relativamente sencillo. Hicimos que el fondo se repitiera horizontalmente con `repeat-x`:

```
body{background-repeat: repeat-x}
```

Attachment

`attachment` permite especificar cómo funciona el fondo cuando el usuario desplaza la página hacia abajo. El funcionamiento por defecto es `scroll`, que hace que la imagen de fondo se desplace junto con el contenido.

Por otra parte, si se establece `background-attachment` en `fixed`, el elemento se queda fijo en la ventana del navegador, de modo que no se mueve cuando se desplaza el contenido de dentro del elemento en el que está adjunto. Esto crea algunos efectos extraños que sólo serán visibles cuando desplacéis la página por encima el elemento de HTML en el que está adjunto. La W3C lo utiliza para señalar el estado de las especificaciones, como por ejemplo la imagen de la "W3C Candidate Recommendation" de arriba del todo a la izquierda en sus especificaciones*. Desplazad la página hacia abajo y veréis que la imagen se queda fija en el rincón superior izquierdo. Está adjunta al elemento `body`, de manera que siempre es visible.

*** http://www.w3.org/TR/CSS21/colors.html#propdef-background-repeat**

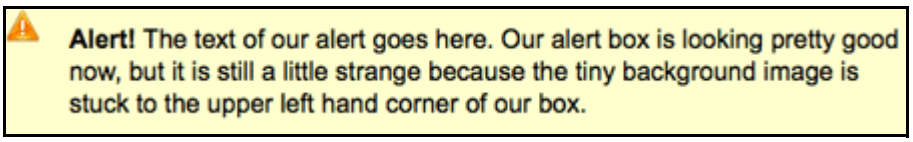
Este paso no tendrá ningún efecto en nuestra visualización porque los navegadores establecen por defecto que las imágenes de fondo se desplacen, pero

añadámoslo igualmente al código para que podáis ver cómo se utiliza la propiedad. Añadid la línea resaltada a la regla de CSS:

```
.alert{
  background-color: #FFFFCC;
  background-image: url(alert.png);
  background-repeat: no-repeat;
  background-attachment: scroll;
}
```

Como se observa en la figura 9, la visualización de la caja de alerta no es muy diferente de como era antes.

Figura 9



No ha cambiado mucho, aún.

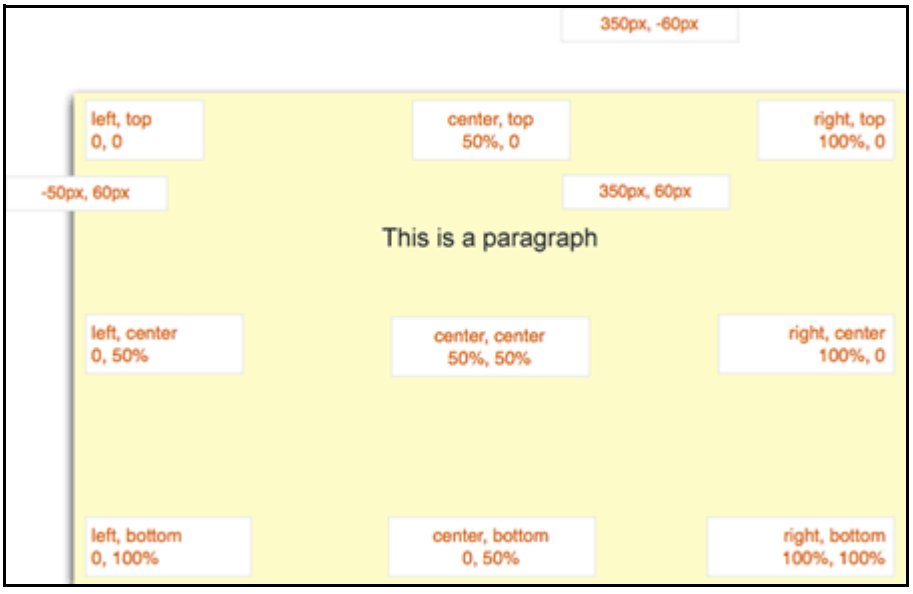
Archivo fuente de: "Figura 9"
http://mosaic.uoc.edu/ac/le/operafiles/31/5_attachment.html

Colocar la imagen

La propiedad de colocación permite disponer la imagen de fondo exactamente donde queráis ponerla, tanto horizontal como verticalmente, en el elemento de HTML. Esta propiedad acepta palabras clave y valores como: `top`, `center`, `right`, `100%`, `-10%`, `50px` y `-30 em`.

En la figura 10 se muestran los valores que pueden utilizarse para colocar las imágenes de fondo en diferentes posiciones.

Figura 10



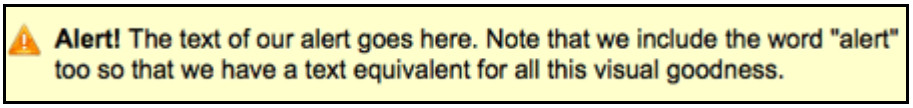
Varios ejemplos de posición del fondo mediante palabras clave, porcentajes y píxeles

Pasemos ahora a colocar la imagen de fondo: queréis que quede en el extremo superior izquierdo, pero sin tocar los lados, de manera que necesitáis establecer una distancia de 10 píxeles tanto desde el límite superior como desde el izquierdo. Esto se puede hacer añadiendo la siguiente línea resaltada en la regla CSS. Hacedlo ahora.

```
.alert{
  background-color: #FFFFCC;
  background-image: url(alert.png);
  background-repeat: no-repeat;
  background-attachment: scroll;
  background-position: 10px 10px;
}
```

El primer valor es la distancia horizontal y el segundo, la vertical. En este caso, son el mismo. Ahora la caja de alerta quedará como en la figura 11:

Figura 11



Hemos utilizado la colocación para situar la imagen de fondo.

Archivo fuente de: "Figura 11"
http://mosaic.uoc.edu/ac/le/operafiles/31/6_position.html

> **Consejo**
>
> Utilizad o bien sólo palabras clave o bien sólo valores numéricos, ya que los navegadores antiguos puede que ignoren vuestra declaración si los utilizáis los dos a la vez. Si utilizáis `right` y `bottom`, conseguiréis lo mismo que `100%` horizontal y verticalmente.

Usar abreviaciones para combinarlo todo como profesionales

Como ya habéis podido observar, algunas propiedades de CSS pueden combinarse. Por ejemplo, el fondo y todas las subpropiedades relacionadas. El código de CSS que hemos escrito hasta ahora se puede reescribir abreviadamente así:

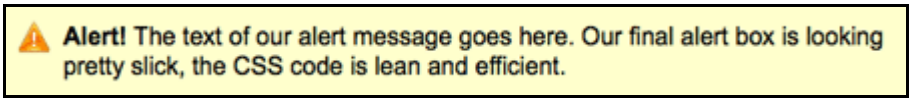
```
.alert {background: #FFFFCC url(alert.png) no repeat scroll 10px 10px;}
```

> **Consejo**
>
> Cuando combinéis subpropiedades de `background`, poned siempre las propiedades en el orden siguiente. Es importante tanto para la compatibilidad de todos los navegadores, como para la organización y el mantenimiento de la hoja de estilo:
>
> 1. `color`
> 2. `image`

3. `repeat`
4. `attachment`
5. posición `horizontal`
6. posición `vertical`

Intentad sustituir el CSS antiguo con la abreviación anterior, y vuestro ejemplo os debería quedar exactamente igual. Podéis ver la figura 12.

Figura 12



La abreviación funciona de maravilla!

Archivo fuente de: "Figura 12"
http://mosaic.uoc.edu/ac/le/operafiles/31/7_finishedProduct.html

5.2.3. Experimentar con el código

La mejor manera de recordar todos los detalles de CSS es probar las opciones uno mismo. Intentad cambiar algunas de las propiedades del ejemplo y ved cómo queda. Estableced la `background-position` a `100% 100%`, y fijaos en que da el mismo resultado que utilizar las palabras clave de `right` y `bottom`. ¿Y si lo cambiarais a `-5px 0`? ¿Por qué creéis que ahora no podéis ver parte de la imagen?

5.2.4. Comprobación de calidad

Hacer pruebas es extremadamente importante para ofrecer una buena experiencia de usuario. El hecho de que el sitio funcione bien en vuestro equipo con vuestra configuración específica no quiere decir que se vea bien en todos los ordenadores. Cuando comprobéis la caja de alerta, deberíais seguir estos pasos básicos mínimos:

- Aumentad o reducid la cantidad de texto en el interior de la alerta.
- Aumentad el tamaño del texto de vuestro navegador como mínimo dos niveles. ¿Habría sido mejor utilizar ems para colocar la imagen? ¿Y qué sucede cuando aumentáis el tamaño del texto?
- Aplicad la alerta `class` a otros elementos como `div`, `p`, `ul`, `strong` o `em`. ¿Qué necesitáis cambiar para hacer que la clase sea agnóstica?
- Incluid unos cuantos párrafos y una lista dentro del `div` de la alerta. ¿Aún funciona el código?
- Verificad la alerta visualmente en los navegadores de grado 1* (también llamado *grado A*). Mi consejo es que escribáis para buenos navegadores y adaptéis para Internet Explorer después de que funcione el código.

*** http://www.bbc.co.uk/guidelines/futuremedia/technical/browser_support.shtml#support_table**

Realizar una serie rigurosa de pruebas forma parte del proceso de aprender a escribir CSS. Si vais con mucho cuidado a la hora de aprender, acabaréis yendo muy rápido.

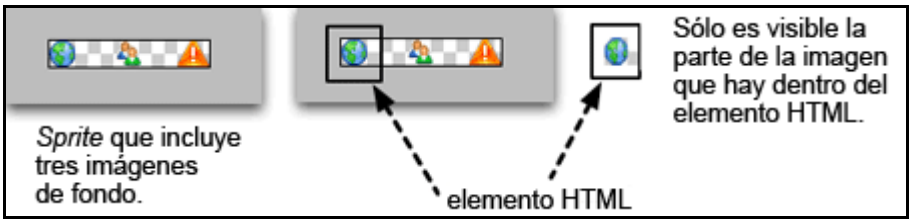
5.3. *Sprites*

Los usuarios lo quieren todo. Quieren que vuestro sitio sea atractivo, interactivo y, encima, rápido, pero incluir grandes cantidades de imágenes de fondo de CSS puede ralentizar mucho la web, ya que cuantas más peticiones HTTP hagáis, más lento será el sitio (una petición HTTP es cuando el ordenador está accediendo a una página web y necesita pedir al servidor que le envíe otro valor que compone el sitio, como un fichero o una imagen de CSS; cada petición adicional significa más tiempo de carga del sitio). Con el fin de superar esta limitación, podéis combinar iconos relacionados para formar una sola imagen, los llamados *sprites* CSS*. La propiedad de `background-position` permite colocar la imagen en la posición adecuada de manera que los iconos se vean a través de la ventana del elemento HTML donde están adjuntos los *sprites* CSS.

* http://www.alistapart.com/articles/sprites

En la figura 13, por ejemplo, observaréis que para ver el icono de la Tierra a través de la ventana HTML debéis colocar la imagen utilizando `left top`. Para mover la posición de la imagen de manera que se vea el icono de alerta, la posición del fondo se debe cambiar a `-80 px 0`. El valor negativo horizontal hace que la imagen se desplace a la izquierda.

Figura 13



Utilización de *sprites* CSS para reducir las peticiones de HTTP

Nota

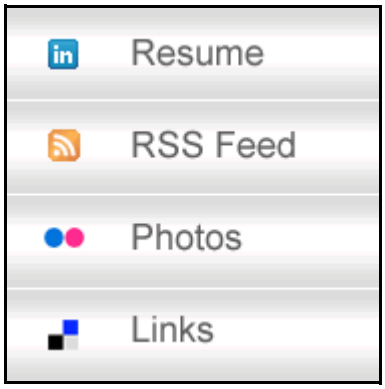
Si utilizáis posiciones de fondo negativas, Safari repetirá la imagen aunque hayáis especificado `no repeat`. Esto hay que tenerlo en cuenta cuando empecéis a jugar con las imágenes de fondo para crear composiciones más complicadas.

5.3.1. Un ejemplo de *sprite* complejo e imagen de fondo

Fijémonos ahora en cómo se pueden utilizar los *sprites* CSS para conseguir buenos efectos. Imaginémonos que nuestro amigo diseñador nos ha enviado otro boceto. En este caso, se trata de una lista de enlaces a la página de destino de

un blog. Conduce al perfil LinkedIn, la suscripción RSS, fotos Flickr y los enlaces favoritos del escritor del blog. Si nos fijamos en cada enlace, nos daremos cuenta de que hay una gradación de color que empieza siendo blanco hasta llegar al gris en la parte superior e inferior del enlace y, para acabar de complicarlo más, el diseñador nos ha pedido si podemos hacer que el fondo de cada enlace sea de color blanco neutro sin gradación cuando los visitantes pasen el cursor por encima; podéis ver la figura 14.

Figura 14



El boceto del nuevo diseño

Los logos podrían incluirse utilizando elementos `img` en el etiquetado, pero utilizar *sprites* CSS es mucho más práctico, porque como sólo hay que cargar una imagen (y no cuatro), los *sprites* se cargan más deprisa, lo que simplifica el HTML y reduce la cantidad de etiquetado necesario.

Crear el *sprite*

El primer paso es recortar los cuatro logos y crear el conjunto de *sprites* como en la figura 15.

Figura 15

El conjunto de *sprites*

También hay que recortar un trozo de la gradación de un píxel de anchura. Para mejorar la visibilidad, hemos recortado un trozo ligeramente mayor, pero vosotros sólo necesitáis un píxel; podéis ver la figura 16.

Figura 16

El trozo de nuestro fondo de gradación

El HTML de la lista es una lista no ordenada de enlaces. Fijaos en los elementos `span` vacíos que hay en los enlaces. Es muy importante no tener una `height` y `width` fija en los elementos que contienen texto, porque al fin y al cabo no sabemos cómo será de grande el texto. ¿Qué sucedería si se tradujera el texto al alemán? Se pueden utilizar estos *spans* adicionales para mostrar los logos.

Alternativamente, podéis decidir que no queréis tener etiquetado no semántico superfluo abarrotando el HTML. En este caso, necesitaréis utilizar un *sprite* mayor y dejar espacio blanco entre los iconos. Tened en cuenta que esto será más lento para los usuarios con conexiones lentas, sobre todo los que acceden a Internet por teléfono móvil. El código de la lista es el siguiente (añadidlo a una plantilla HTML):

```
<ul class="navigation">
  <li id="resume">
    <a href="#"><span></span>Resume</a>
  </li>
  <li id="rss">
    <a href="#"><span></span>RSS Feed</a>
  </li>
  <li id="photos">
    <a href="#"><span></span>Photos</a>
  </li>
  <li id="links">
    <a href="#"><span></span>Links</a>
  </li>
</ul>
```

El CSS utiliza las dos imágenes de fondo. Primero, fijaos en la gradación de la imagen de fondo. Tiene tres aspectos interesantes:

1) El primero es que la imagen se repite horizontalmente (`repeat-x`). De esta manera podemos hacer que una imagen tan pequeña se extienda por toda la lista.

2) El segundo es que la imagen está centrada verticalmente. El trozo redondo de la imagen debe quedar en medio del elemento de la lista, de modo que debéis utilizar una posición de fondo de `left center`.

3) Para acabar, en el CSS hemos aplicado un color de fondo que es el mismo gris que el de nuestra imagen de gradación. Así, si el elemento se hace más grande, no quedará roto. Para saber más sobre esta técnica, recomendamos leer *Bulletproof Web Design**, de Dan Cederholm.

*** http://www.simplebits.com/publications/bulletproof/**

Añadid el siguiente CSS a un nuevo fichero CSS y enlazadlo al fichero HTML:

```
.navigation, .navigation li {
  margin:0;
  padding:0;
}
```

```
.navigation li {
  border-top: 1px solid white;
  list-style-type:none
}
.navigation li a {
  background: #E2E2E2 url(sprite_gradient_bkg.jpg)
repeat-x left center;
  padding:20px;
  display:block;
  font-family: Arial, Helvetica, sans-serif;
  color:#333;
  font-size:18px;
  text-decoration:none
}
/* hover effects */
.navigation li a:hover, .navigation li a:focus{
  background: transparent none;
}
```

La última línea significa que el elemento no debería tener color ni imagen de fondo cuando el usuario pase el ratón por encima o lo seleccione con el teclado. Puede que os estéis preguntando por qué hemos aplicado propiedades de fondo al enlace y no al elemento de la lista. La respuesta es que Internet Explorer 6 y las versiones anteriores no reconocen pseudoclases como `hover` en elementos que no sean enlaces. Hemos hecho el ajuste para adaptarnos a esta limitación.

A continuación, podéis crear el CSS para los logos pequeños. Como siempre, podéis empezar definiendo el caso más general para todos los elementos `span` de vuestro módulo de navegación. Es aquí donde definís la imagen que se utilizará en todos los *spans*, la repetición y la posición del fondo (cada uno es diferente, de manera que utilizaremos la primera). Para esta regla podéis utilizar la abreviación. Fijaos en que utilizamos comentarios de CSS para dividir secciones del código en trozos más prácticos. Añadid el siguiente código al final del fichero CSS:

```
/* caso general */
.navigation span {
  background:url(sprite_logo.gif) no-repeat left top;
  height:15px;
  width: 15px;
  margin-right:20px;
  display:-moz-inline-box;
  display:inline-block;
  vertical-align: middle;
}
```

Una vez bien definido el caso general, ya podéis pasar a definir las excepciones, o lo que tiene de diferente cada logo en concreto. En este caso, el único CSS que cambia es la `background-position`. Cada elemento de la lista respectivo necesita tener la imagen 15 píxeles más hacia la izquierda porque cada logo tiene 15 píxeles de anchura. Añadid el siguiente código al final del fichero CSS:

```
/* exceptions */
#rss span {
  background-position: -15px 0;
}
#photos span {
  background-position: -30px 0;
}
#links span {
  background-position: -45px 0;
}
```

Este ejemplo puede intimidar un poco al principio, pero centraos en las imágenes de fondo. En este caso hemos utilizado valores de píxel negativos para mover la imagen de fondo hacia la izquierda de manera que se vea la parte relevante de la imagen. Los valores positivos mueven la imagen de fondo hacia abajo y hacia la derecha, mientras que los negativos mueven la imagen hacia arriba y hacia la izquierda.

Haced pruebas con los valores de posición de la imagen de fondo con el ejemplo acabado para entender mejor cómo ajustar la colocación de los *sprites*.

Ved el ejemplo acabado en: "*Sprite* example"
http://mosaic.uoc.edu/ac/le/operafiles/31/sprite.html

Resumen

Ahora ya deberíais entender las imágenes de fondo de CSS y, además, ya estaréis adquiriendo práctica en leer las especificaciones, de manera que si tenéis alguna duda sobre alguna propiedad en particular, deberíais saber cómo consultarla. En este apartado hemos hablado del color de fondo, la imagen, la repetición, la adjunción y la posición. También habéis aprendido por qué los desarrolladores utilizan *sprites* de CSS y cómo utilizar esta técnica avanzada.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. Un párrafo tiene unas dimensiones de 40 píxeles por 180 píxeles y la imagen de fondo es de 60 píxeles por 200 píxeles. ¿Veréis toda la imagen o sólo una parte? ¿Por qué?

2. Debéis hacer que una imagen quede colocada en el extremo inferior izquierdo del elemento `blockquote`. Llenad los espacios con los valores correctos.

```
blockquote{background: yellow url(quote.png) no repeat
scroll ____ ____;}
```

3. Pongamos por caso que queréis que a cada `h2` de vuestro documento con una `class` de "pregunta" se le aplique un patrón de gradación. ¿Utilizaríais `repeat-x`, `repeat-y`, `no-repeat` o `repeat` para conseguir un resultado parecido al ejemplo siguiente? ¿Por qué?

4. ¿Cuál sería la posición del fondo del ejemplo de la pregunta 3? ¿Cómo podríais utilizar un color de fondo de manera creativa para garantizar que el fondo pueda ampliarse a cualquier altura? ¿Por qué es esto importante?

5. ¿Qué abreviación podéis utilizar para eliminar todas las propiedades de fondo?

6. ¿Para qué sirven los *sprites* CSS?

Lecturas complementarias

Rendimiento y peticiones HTTP en la *Yahoo Developer Network*
http://developer.yahoo.com/performance/rules.html#num_http

La especificación CSS2-Colores y fondos
http://www.w3.org/TR/REC-CSS2/colors.html

CSS Sprites-Image Slicing's Kiss of Death
http://www.alistapart.com/articles/*sprites*

Bulletproof Web Design-Mi libro favorito
http://www.simplebits.com/publications/bulletproof/

Créditos de las imágenes

The Tiles, DimsumDarren
http://www.flickr.com/photos/dimsumdarren/1342305614/

little glass tiles, emdot
http://www.flickr.com/photos/emdot/6099842/

6. Estilos de listas y enlaces

Ben Buchanan

Muchos elementos de las páginas web son un tanto "indulgentes" en cuanto al diseño: no es muy grave si no están "del todo bien". Pero listas y enlaces son otra historia, porque si no se hacen bien, pueden causar problemas graves a la gente que quiera utilizar vuestro sitio web.

En concreto, los enlaces tienen algunos requisitos básicos en cuanto al estilo y a las expectativas de los usuarios. Los enlaces con estilos mal hechos pueden arruinar la experiencia que tendrá el usuario cuando utilice la web porque la gente deberá detenerse a pensar dónde debe hacer clic. En el peor de los casos, el usuario no sabrá determinar ni siquiera qué elementos de la página son enlaces.

En este apartado, nos centraremos en las competencias básicas necesarias para crear unos estilos de listas y de enlaces bien hechos. También hablaremos de algunas maneras de evitar errores fundamentales de estos elementos y de producir un resultado final que funcione en cualquier navegador y que sea accesible para los usuarios con discapacidades.

> **Nota**
>
> En este apartado se utilizarán varios ejemplos, de modo que os podéis descargar los ficheros de muestra de las listas y los enlaces para ir viéndolos a medida que sigáis la lección.

Ved ejemplos reales en: "styling-lists-and-links examples"
http://mosaic.uoc.edu/ac/le/operafiles/32/styling-lists-and-links-examples.zip

6.1. Aplicar estilos a las listas

Antes de nada, os enseñaremos los conceptos básicos relacionados con la aplicación de estilos a las listas con CSS y después pasaremos a echar un vistazo a algunas de las técnicas algo más complicadas.

6.1.1. Picos y números básicos

El punto más básico que se debe tener en cuenta a la hora de crear un estilo para una lista es pensar en qué forma de pico o de número queremos utilizar. También puede ser que no queráis utilizar ningún tipo de pico o de número. Como habéis aprendido al ver el HTML para las listas, existen muchas posibilidades que se establecen mediante la propiedad `list-style-type`.

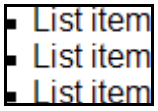
Podéis ver las listas en el apartado 2 del módulo "El texto central de HTML".

Por ejemplo, para especificar que todas las listas no ordenadas del sitio web utilicen picos cuadrados, podéis utilizar este CSS:

```
ul le {
  list-style-type: square;
}
```

De esta manera, conseguiremos un resultado parecido al de la figura 1:

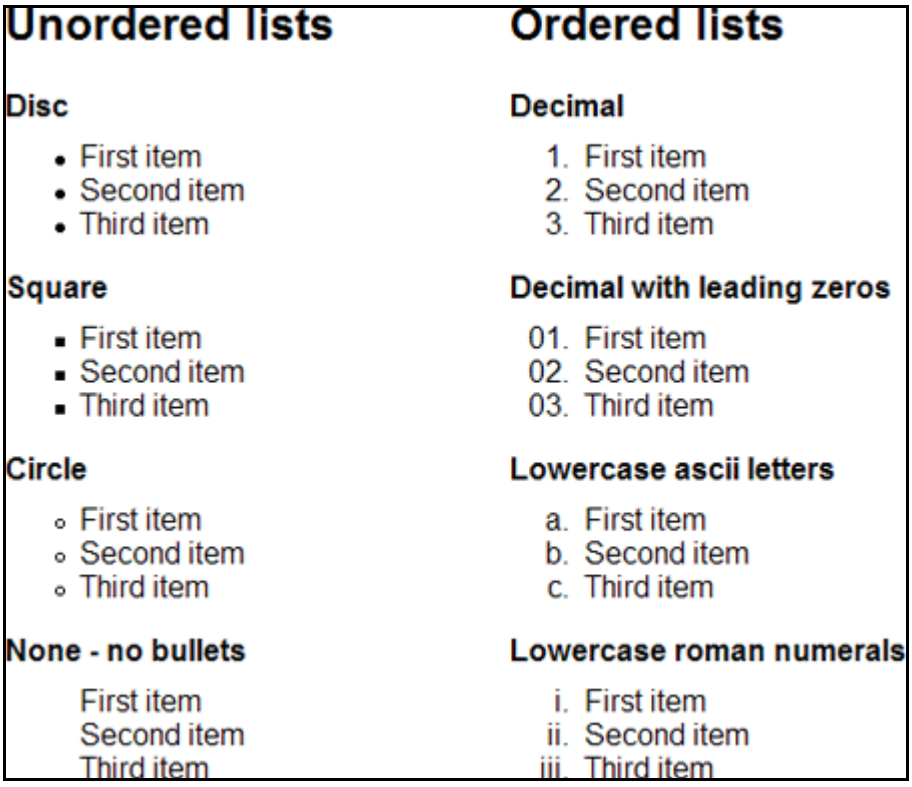
Figura 1



Lista desordenada con picos cuadrados

En la figura 2 se muestran algunos ejemplos de lista muy habituales:

Figura 2



Estilos más habituales de listas

Podéis ver algunas opciones más en la página de ejemplos en: "styling-lists-example-basics"
http://mosaic.uoc.edu/ac/le/operafiles/32/styling-lists-example-basics.html

Fijaos en que los picos y los números se representan con el `color` que se especifique en `li` o que éste herede. Si necesitáis que el pico sea de un color diferente del texto, deberéis utilizar una imagen o solucionarlo utilizando otros elementos de los puntos enumerados en la lista (puede ser muy fácil si la lista está formada por enlaces, por ejemplo).

6.1.2. Picos personalizados mediante imágenes

El conjunto de picos estándar es suficiente para el contenido básico, pero a menudo se pide a los diseñadores que sustituyan los picos por una imagen propia.

La especificación de CSS incluye la propiedad `list-style-image` para añadir una imagen de lista personalizada. No obstante, la propiedad tiene pocas opciones de colocación de la imagen de fondo y, en determinadas circunstancias, no funciona en absoluto en IE. Por ello, ha acabado siendo mucho más habitual establecer una imagen de fondo en los elementos enumerados en la lista.

Pongamos por caso que tenéis una lista de canales de información RSS y queréis cambiar el pico por el típico icono naranja de RSS. Aplicaremos la clase "rss" a la lista para diferenciarla de otras listas.

```
<ul class="rss">
  <li><a href="http://example.com/rss.xml">News</a></li>
  <li><a href="http://example.com/rss.xml">Sport</a></li>
  <li><a href="http://example.com/rss.xml">Weather</a></li>
  <li><a href="http://example.com/rss.xml">Business</a></li>
  <li><a href="http://example.com/rss.xml">Entertainment</a></li>
  <li><a href="http://example.com/rss.xml">Funny News</a></li>
</ul>
```

Primero estableceremos que la lista no tenga `list-style-type` y eliminaremos el margen y el relleno (*padding*). Entonces, sólo hay que añadir una imagen de fondo en cada elemento enumerado de la lista, algo de relleno a la izquierda para desplazar el texto para que se pueda ver la imagen y algo de relleno alrededor del botón para espaciar los elementos de la lista.

```
.rss {
  margin: 0;
  padding: 0;
  list-style-type: none;
}
.rss li {
  background: #fff url("icon-rssfeed.gif") 0 3px no-repeat;
  padding: 0 0 5px 15px;
}
```

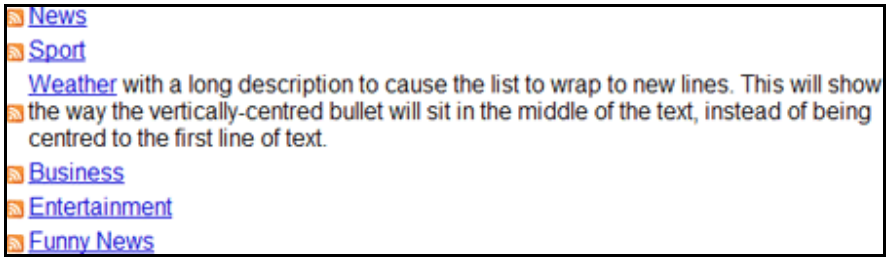
De esta manera, nos quedará una lista con la imagen de RSS en lugar de picos, tal como se puede observar en la figura 3:

Figura 3

Lista con picos hechos con una imagen

Tened en cuenta que la imagen de fondo está situada utilizando píxeles para colocarla de manera precisa. Dependiendo del diseño que queráis aplicar, puede ser que también podáis utilizar `%`, `em` o palabras clave. De todos modos, id con cuidado cuando el diseño incluya contenido que pueda hacer que un elemento de la lista acabe teniendo varias líneas de texto, porque si establecéis el fondo en `center` o `50%` verticalmente, puede quedar bastante extraño, como se ve en la figura 4:

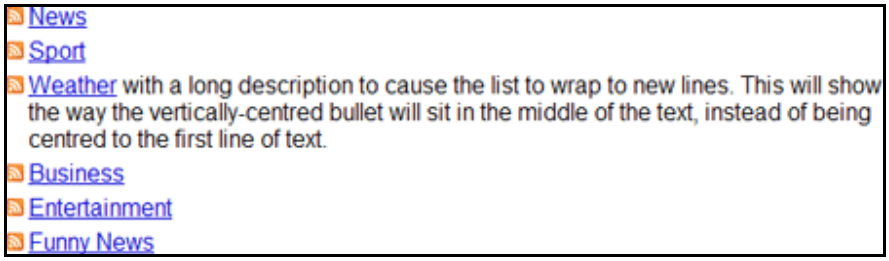
Figura 4



Demostración de imágenes de pico centradas verticalmente en un elemento de lista de varias líneas

En cambio, si establecéis que la imagen quede en la parte superior del elemento de la lista, conservaréis el comportamiento tradicional de los picos (es decir, quedará delante de la primera línea). Podéis ver la figura 5:

Figura 5



Demostración de imágenes de pico alineadas en la parte superior de un elemento de lista con más de una línea

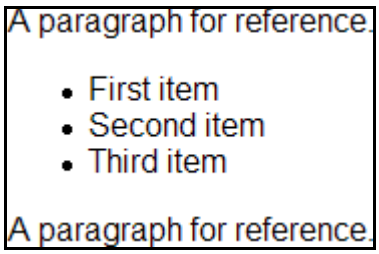
6.1.3. Márgenes y relleno de listas

Si se utilizan bien, los márgenes y el relleno pueden hacer que las listas queden mucho más ordenadas y profesionales, pero debéis saber qué estáis haciendo y tener en cuenta que la situación es diferente para cada tipo de lista. En este subapartado os enseñaremos cómo aplicar márgenes y relleno a los dos tipos de lista más habituales, de manera juiciosa.

Listas no ordenadas

Algo que probablemente notaréis enseguida es que el estilo predeterminado de las listas aplica un sangrado más destacado que el estilo predeterminado para los párrafos. Podéis ver la figura 6:

Figura 6



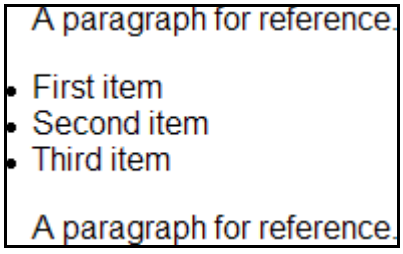
Las listas con el estilo predeterminado tienen un sangrado a la izquierda.

Si queréis que los elementos de una lista no ordenada queden alineados en el mismo punto que el resto de contenido, deberéis crear algunos estilos para modificar el sangrado como más os guste. Cada navegador requiere unos parámetros diferentes, en algunos se debe eliminar el margen y en otros se debe eliminar el relleno. Por lo tanto, para establecer los valores iniciales en todos los navegadores, estableced lo siguiente:

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

Puede que eso no tenga el efecto que esperabais porque hará que todo el texto quede a la misma altura tocando a la izquierda, pero los picos quedarán más allá del texto, como se ve en la figura 7:

Figura 7



Los picos quedan más a la izquierda que el texto.

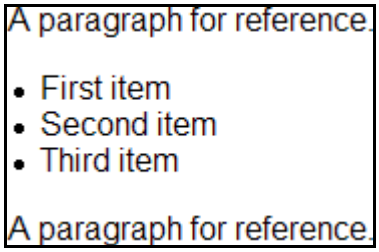
Por lo tanto, para alinear los picos a la izquierda, lo que debéis hacer es establecer un margen para la lista de elementos y que así quede todo al mismo nivel:

```
ul {
  margin-left: 0;
  padding-left: 0;
}
```

```
ul le {
  margin-left: 1em;
}
```

Llegados a este punto, aún encontraréis alguna diferencia de un píxel entre un navegador y otro, pero el efecto es básicamente el más consistente posible. Podéis ver la figura 8:

Figura 8

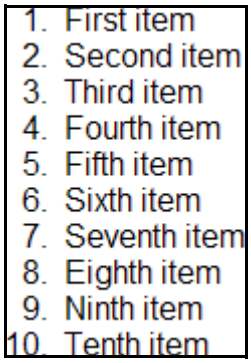


Los picos colocados al mismo nivel que los párrafos anteriores y posteriores

Listas ordenadas

Ahora debéis tener en cuenta lo mismo, pero aplicado a las listas ordenadas. Estas listas son más complicadas porque los símbolos numéricos están alineados según el elemento de la lista que tenga el número más alto. Por ejemplo, si tenéis una lista de 10 elementos, los decimales se colocarán de manera que el elemento de dos dígitos, el "10", quede como se ve en la figura 9:

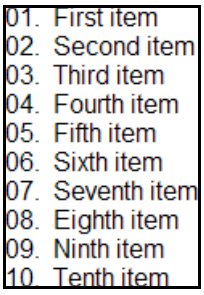
Figura 9



Los símbolos numéricos de los elementos del 1 al 9 tienen un relleno a la izquierda para quedar alineados con el elemento 10.

Por lo tanto, en realidad no se puede hacer que esta lista quede toda alineada a la izquierda y al mismo nivel que el resto del texto, a no ser que establezcáis que la lista utilice `list-style-type: decimal-leading-cero;`, que solucionará el problema tal como se ve en la figura 10:

Figura 10



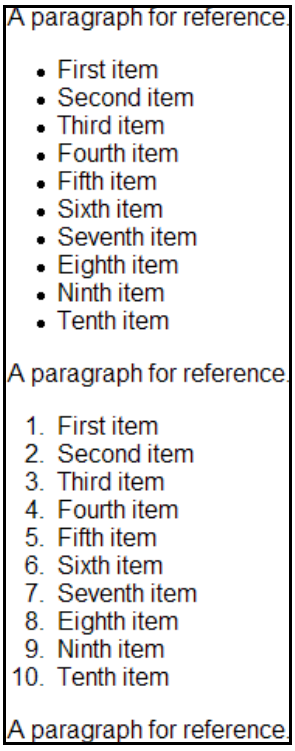
Los ceros a la izquierda llenan el espacio vacío de los elementos del 1 al 9.

Es más habitual aceptar sencillamente la diferencia de espaciado. Sin embargo, esto significa que los símbolos de las listas ordenadas y desordenadas no podrán alinearse fácilmente a la izquierda de manera consistente. Sólo se podrá alinear el texto de las listas.

```
ul, ol {
  margin-left: 0;
  padding-left: 0;
}
li {
  margin-left: 2em;
}
```

Necesitáis como mínimo 2 em de margen a la izquierda para incluir tanto las listas ordenadas como las no ordenadas. En la figura 11, fijaos en cómo se alinea el texto de los elementos en las dos listas.

Figura 11



El texto se alinea tanto en las listas ordenadas como en las desordenadas.

¿Entonces qué hay que hacer?

Básicamente tenéis tres opciones:

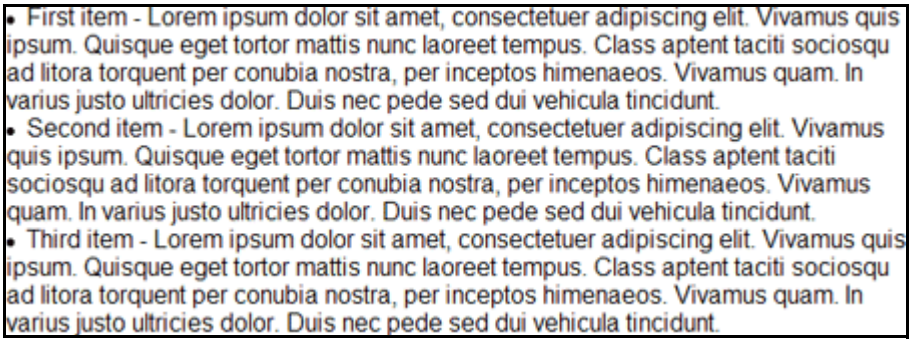
1) Aceptar la colocación por defecto de las listas y de sus símbolos.
2) Especificar la alineación del texto de las listas.
3) Establecer un estilo diferente para `ul` y `ol`.

No hay una manera correcta ni incorrecta de hacerlo y es bastante habitual que se deje todo con la configuración predeterminada para las listas de contenido general.

6.1.4. Utilizar una colocación de tipo lista

Si queréis que el texto de elementos de más de una línea quede por debajo del símbolo de la lista, deberéis configurar la propiedad `list-style-position` en `inside` para así obtener el resultado que se ve en la figura 12:

Figura 12



La posición de la lista `inside` hace que, a partir de la segunda línea, el texto quede por debajo del símbolo en lugar de alineado con el sangrado del texto.

Este estilo de símbolos rodeados de texto no es muy popular. La propiedad `list-style-position` está establecida por defecto en `outside`, lo que produce el resultado del que se habla más adelante en este apartado.

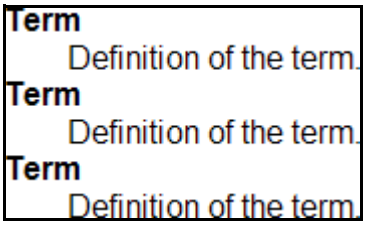
6.1.5. ¿Y las listas de definiciones, qué?

En general, las listas de definiciones no requieren mucha atención aparte de establecer un estilo `dt` (normalmente texto en negrita) y controlar el sangrado de las definiciones:

```
dt {
  font-weight: bold;
}
dd {
  margin-left: 2em;
}
```

Eso define un estilo claro y sencillo para listas de definiciones, como se ve en la figura 13:

Figura 13



Una lista de definiciones con un estilo muy sencillo

Aunque las listas de definiciones pueden reestructurarse con elementos flotantes y posicionamientos, esto es complicado y, en general, es mejor hacer las cosas sencillas. Como ya son bastante útiles, sólo les hace falta un poco de ayuda para hacer que los términos de la definición destaquen un poco más y que las definiciones tengan un sangrado adecuado.

6.1.6. Listas anidadas

Ya aprendisteis a anidar listas. Cuando creéis vuestro propio CSS, debéis ir con cuidado de mantener unos rasgos de diseño muy claros para mostrar la relación entre una lista anidada y la lista que la incluye. La manera más fácil de conseguirlo es aplicar un sangrado a los elementos de la lista anidada que, de hecho, es la configuración predeterminada en todos los navegadores.

Podéis ver las listas en el subapartado 2.4 del módulo "El texto central de HTML".

Si establecéis vuestro sangrado de lista personal, la configuración básica se multiplicará. Por ejemplo, podéis ver el CSS siguiente:

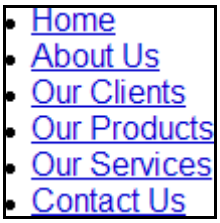
```
ul, ol {
  margin-left: 0;
  padding-left: 0;
}
li {
  margin-left: 2em;
}
```

Cada elemento subsiguiente de la lista hija de la cadena hereda el valor de `margin` de su elemento de lista padre además de tener 2 em propios más añadidos encima. Así pues, un elemento de lista de nivel superior (uno que no tenga un elemento de lista como elemento padre) tendrá un margen izquierdo de 2 em, mientras que un elemento de lista hijo del primer elemento de lista heredará 2 em de su padre y tendrá 2 em más añadidos encima, es decir, un total de 4 em, etc.

6.1.7. Listas horizontales

Uno de los cambios más comunes que son necesarios para trabajar con una lista es crear una lista horizontal, es decir, hacer que los elementos de la lista aparezcan uno al lado del otro en vez de uno debajo del otro. Se trata de un truco muy habitual para la navegación por el sitio. Veamos un ejemplo del apartado de los menús de navegación (figura 14):

Figura 14



Una lista sencilla

Convirtámosla en una lista horizontal como se ve en la figura 15:

Figura 15

Una lista horizontal sencilla

Para conseguir este efecto, hemos de hacerle tres cosas a la lista:

1) Eliminar `margin` y `padding` de `<ul>`.
2) Establecer los elementos de la lista en `display: inline;`.
3) Aplicar un poco de espaciado a la derecha de los elementos de la lista para que no choquen.

En este ejemplo, la lista tiene el ID "mainmenu" (menú principal), de manera que lo utilizaremos como selector contextual para garantizar que sólo cambiamos la lista que queremos cambiar. El CSS es el siguiente:

```
#mainmenu {
  margin: 0;
  padding: 0;
}
#mainmenu {
  display: inline;
  padding: 0 1em 0 0;
}
```

En este sencillo ejemplo sólo hay que establecer los elementos de la lista en `display: inline;`. Tened en cuenta que si utilizáis `float: left;` también conseguiréis un resultado parecido. Más adelante aprenderéis más cosas sobre los elementos flotantes.

Podéis ver los elementos flotantes en el apartado 9 de este módulo didáctico.

6.1.8. "Columnas falsas"

Antes hemos creado una lista de canales de información de RSS. Ahora imaginaos que aquella lista se ha situado en una barra lateral de vuestro sitio. El diseñador quiere que la lista aparezca en dos columnas con un borde alrededor de todo el grupo como se ve en la figura 16:

Figura 16

Una lista de canales de información en dos columnas con un icono de RSS como pico

Ved el ejemplo en: "Styling lists example columns"
http://mosaic.uoc.edu/ac/le/operafiles/32/styling-lists-example-columns.html

Supongamos que la lista se encuentra dentro de un `<div>` que establece la anchura y el borde. La lista básica quedaría de una manera parecida a la figura 17:

Figura 17

La lista sin estilos dentro del borde

Ved la lista básica original en: "Styling lists example image bullet"
http://mosaic.uoc.edu/ac/le/operafiles/32/styling-lists-example-basics.html

Para conseguir el efecto de "faux columns" o columnas falsas, añadid el icono RSS como hemos visto antes y aplicad un margen de 5 píxeles arriba, a la derecha y a la izquierda.

```
.rss {
  margin: 5px 5px 0 5px;
  padding: 0;
}
.rss li {
  list-style-type: none;
  background: #fff url("icon-rssfeed.gif") 0 3px no-repeat;
  padding: 0 0 5px 15px;
  display:-moz-inline-box;
}
```

Nota

Fijaos en que se añade `display:-moz-inline-box;` para que el ejemplo se pueda ver correctamente con Firefox 2.

No es necesario que le añadamos margen al botón porque el último elemento de la lista ya aplicará el espaciado adecuado con su relleno tal como se observa en la figura 18:

Figura 18

Ya casi hemos terminado; ahora tenemos un espaciado y unos iconos de pico bien hechos.

Ahora estableced los elementos de la lista en `display: inline-block;` y estableced la anchura a `40%` y el margen derecho a `2%` (también podéis utilizar anchuras en píxeles). También debéis especificar que el `<ul>` tenga una anchura de `100%` para garantizar que los elementos de más de una línea queden como deben quedar y que la lista tenga el tamaño correcto:

```
.rss {
  margin: 5px 5px 0 5px;
  padding: 0;
  width: 100%;
}
.rss li {
  display: inline-block;
  width: 40%;
  margin: 0 2% 0 0;
  list-style-type: none;
  background: #fff url("icon-rssfeed.gif") 0 3px no-repeat;
  padding: 0 0 5px 15px;
  display:-moz-inline-box;
}
```

En la mayoría de los navegadores, con esto será suficiente para crear el efecto de la columna, pero deberéis especificar que IE haga que "floten" los elementos de la lista a la izquierda. Utilizamos un estilo condicional para todas las versiones hasta IE 7.

```
<!--[if lte IE 7]>
  <style type="text/css">
    .rss li {
      float: left;
    }
```

```
    </style>
<![endif]-->
```

Ahora ya tenemos el efecto de las dos columnas, tal como se ve en la figura 19:

Figura 19

La lista completa

Navegadores antiguos

Si se os pide que creéis este diseño para navegadores más antiguos que no aceptan `inline-block` (bloque en línea), deberéis hacer que "floten" los elementos de la lista a la izquierda en todos los navegadores y utilizar un ajuste de distancia como el que se describe en el artículo "Clearing a float container without source markup"*. Por suerte, la última hornada de navegadores han hecho que el `inline-block` sea una propiedad de visualización viable, así que, a no ser que tengáis una cuota muy grande de navegadores antiguos como Firefox 2, deberíais poder utilizar el método `inline-block`.

*** http://www.positioniseverything.net/easyclearing.html**

6.1.9. Conclusión de las listas

Ya hemos cubierto un conjunto básico de opciones de aplicación de estilos y métodos para listas. Podéis utilizar estos ejemplos como punto de partida y combinarlos para crear un gran número de diseños. Como las listas a menudo se combinan con enlaces, pasamos a hablar de estos.

6.2. Aplicar estilos a los enlaces

Aplicar estilos a los enlaces puede considerarse una especie de arte. En este campo hay muchos requisitos en juego y puede ser difícil cumplirlos todos y, a la vez, obtener un resultado estéticamente satisfactorio. Por otra parte, es perfectamente posible siempre y cuando se tengan en cuenta una serie de normas muy sencillas:

- comprender los diferentes estados de los enlaces,
- no apartarse mucho de las expectativas del usuario,
- utilizar los colores adecuadamente.

Si seguís estas reglas, podréis crear enlaces claros y fáciles de utilizar.

6.2.1. Comprender los estados de los enlaces

Antes de aplicar estilos a los enlaces, debéis comprender los diferentes **estados de los enlaces**. Hay un total de cinco estados: no visitado o predeterminado, visitado, seleccionado, por encima y activo.

- No visitado (*unvisited*). El estado por defecto de un enlace cuando todavía no se ha activado o visitado.

- Visitado (*visited*). El estado de un enlace que el usuario ya ha visitado.

- Seleccionado (*focus*). Se aplica mientras el enlace está seleccionado; por ejemplo, cuando el cursor del usuario del teclado se encuentra sobre el enlace.

- Por encima (*hover*). Se aplica cuando un usuario tiene el puntero del ratón por encima del enlace pero todavía no ha hecho clic sobre él.

- Activo (*active*). Se aplica cuando el usuario activa el enlace, es decir, mientras está haciendo clic sobre él. En algunos navegadores, este estilo también se aplica cuando el enlace se ha abierto en otra ventana o pestaña.

Siempre debéis especificar el CSS para cada uno de estos estados. Cada uno aporta información al usuario sobre el hecho de interactuar con un enlace. En caso de duda sobre si utilizar `focus`, `hover` o `active`, podéis aplicar el estilo `focus` y `hover` de la misma manera porque sus funciones son bastante parecidas para no causar confusión con el mismo estilo de enlace. Después podéis añadir alguna variación sencilla para `active`, por ejemplo poner el texto en cursiva. En caso de que sea necesario, podéis aplicar el mismo estilo a los tres.

> **Nota**
>
> Fijaos en que no todos estos estados son mutuamente excluyentes (aunque en realidad no es posible que un enlace esté visitado y no visitado a la vez); es perfectamente posible que un enlace esté en estado "por encima", "activo" y "visitado" a la vez.

6.2.2. Cómo la evolución de los navegadores creó expectativas

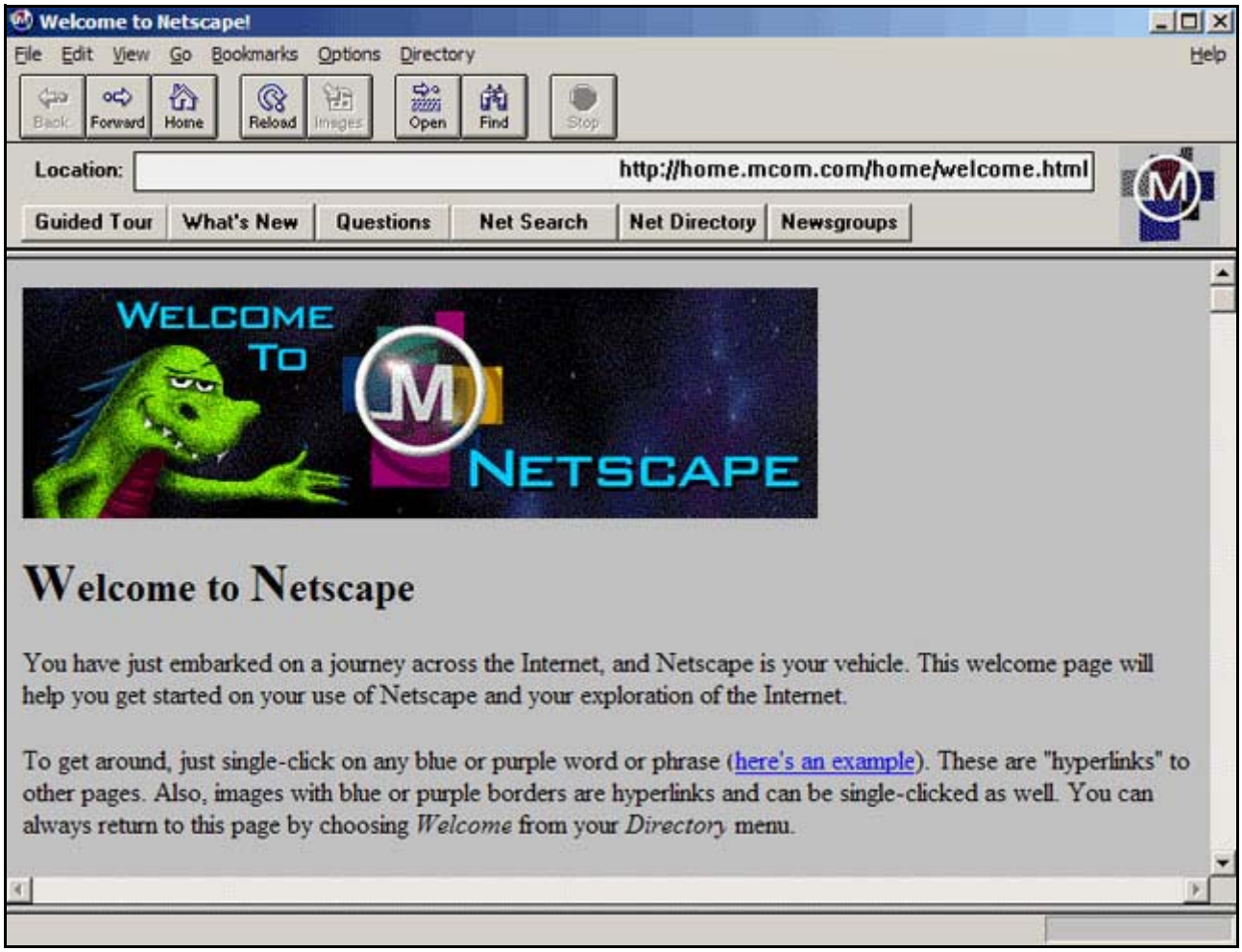
Para entender mejor algunas expectativas de los usuarios respecto a los enlaces, va bien saber un poco de la historia de la web.

Puede que hayáis oído hablar de los "valores por defecto de Netscape" de los enlaces o que los enlaces siempre deberían ser azules y morados. Esto se re-

monta a los principios de la creación de la web, cuando los navegadores establecían los colores del contenido y los creadores de webs no tenían demasiado control sobre la manera de representación de sus páginas.

El texto era negro, el fondo era gris y todos los enlaces estaban subrayados. Los enlaces sin visitar eran azules, los enlaces visitados eran morados y los enlaces activos eran rojos. Esto era todo. Podéis ver la figura 20 para observar una muestra.

Figura 20



Una captura de pantalla de Mosaic

Aunque era un poco monótono, era totalmente consistente y estableció **la base de las futuras expectativas de los usuarios**. Concretamente, los usuarios todavía a día de hoy esperan que el texto subrayado sea un enlace. Quizá no se esperen que todos los enlaces estén subrayados, pero sí esperan que se pueda hacer clic sobre un texto subrayado. Y es mejor no ir en contra de esta expectativa.

Algunos sitios utilizan enlaces azules y morados, que aún son los colores predeterminados del contenido sin estilo en la mayoría de los navegadores. Aunque siempre podéis adaptaros a la moda "retro" y seguir con esta pareja de colores, en general a los usuarios no les importa encontrarse con otros colores, dentro de ciertos límites.

6.2.3. Expectativas del usuario

Hay una serie de reglas generales para las expectativas del usuario en cuanto a los enlaces.

- Los usuarios esperan encontrar enlaces que tengan **un aspecto diferente** del resto del texto que no son enlaces.
- Los usuarios esperan que los enlaces **reaccionen** de alguna manera cuando se pasa el cursor por encima o cuando se selecciona el enlace.
- Los usuarios esperan que los enlaces **cambien** después de visitarlos.
- A los usuarios les gusta la **consistencia** de los estilos de los enlaces de la misma función para así saber sobre qué deben hacer clic.
- Los usuarios esperan que el texto subrayado sea un enlace, así que no subrayéis otra cosa.

Siempre debéis cumplir estas normas básicas porque contribuyen a que los usuarios puedan identificar y utilizar los enlaces rápidamente. Debéis crear estilos que no obliguen a la gente a dudar de qué partes del texto son enlaces.

Estas expectativas se traducen en unas normas de codificación muy sencillas:

- definir estilos para todos los estados de enlace,
- utilizar el subrayado sólo para los enlaces.

6.2.4. Utilización adecuada de los colores

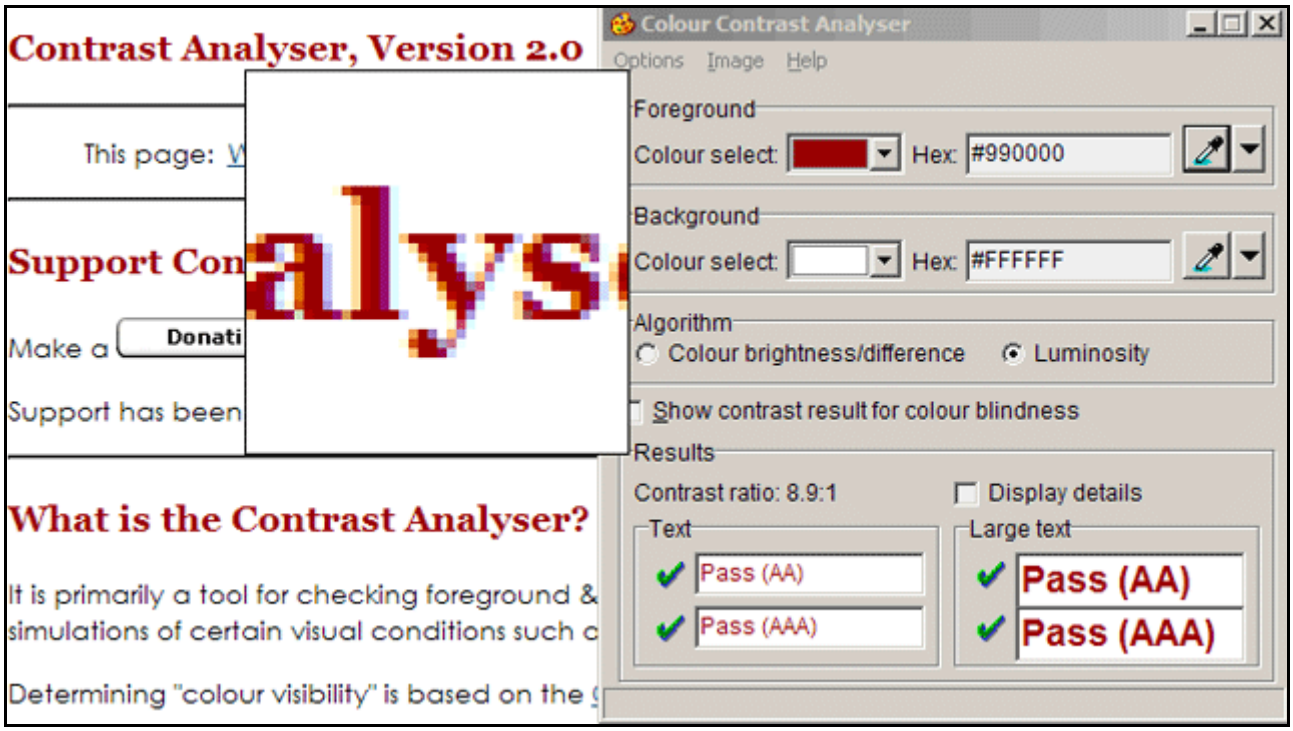
Cuando apliquéis estilos a los enlaces, id con cuidado de no basaros únicamente en el color para distinguir los diferentes estados de enlace. No todo el mundo puede ver igual los colores (como los daltónicos), de manera que debéis utilizar colores y también estilos como diferentes subrayados, iconos o colores invertidos.

También deberíais aseguraros de que los colores que elegís tengan bastante contraste. Es muy fácil si se utilizan herramientas como Colour Contrast Analyser (tanto para PC como para Mac)* o la barra de herramientas de accesibilidad web para Opera** (ambas de Paciello Group).

* http://www.paciellogroup.com/resources/contrast-analyser.html
** http://www.paciellogroup.com/resources/wat-about.html

Con el Colour Contrast Analyser (podéis ver la figura 21) se puede utilizar un selector de colores para seleccionar los colores de letra y de fondo de la pantalla y después obtener una evaluación sencilla del contraste que hacen:

Figura 21



Captura de pantalla del Colour Contrast Analyser

Si los cuatro resultados son "*pass*" (superado), la combinación de colores es adecuada. No os olvidéis de comprobar todos los estados de enlace. Puede que debáis introducir algunos manualmente en el campo "hex" para comprobar el estado "seleccionado", "por encima" y "activo".

6.2.5. Entremos en materia: el CSS

Ahora que ya entendéis las normas básicas de los enlaces, pasemos a ver el código. En este subapartado se describe todo el CSS que necesitaréis para aplicar estilos como es debido.

Aplicar estilos de estado de enlace en el orden adecuado

Para empezar, tened en cuenta que si no aplicáis los estilos a los enlaces en el orden correcto en la hoja de estilo, las definiciones se anularán entre ellas y los estados de los enlaces no funcionarán. Los estilos de los enlaces siempre deben ir en este orden:

1. `link`
2. `visited`
3. `focus`
4. `hover`
5. `active`

Un truco mnemotécnico para recordarlo podría ser: "Los Visitantes Fueron Hasta Allí". Si creéis que no os acordaréis de esta frase, deberéis recordar el or-

den de las propiedades de memoria o copiar y pegar el bloque de código que se muestra a continuación.

Los estilos de los diferentes estados de los enlaces se aplican utilizando sus "pseudoclases" (`:link`, `:visited`, `:focus`, `:hover`, `:active`) que se adjuntan al selector de elemento de enlace, `a`. Por lo tanto, el CSS inicial debería ser el siguiente:

```
a:link{}
a:visited{}
a:focus{}
a:hover{}
a:active{}
```

Si queréis definir una regla de CSS para todos los enlaces en todos los estados, podéis aplicar el estilo `a` directamente. Sólo debéis acordaros de colocar primero la regla genérica para mantener el orden:

```
a {}
a:link{}
a:visited{}
a:focus{}
a:hover{}
a:active{}
```

Eso va muy bien si tenéis pensado sustituir el subrayado por defecto por un borde inferior, que es lo que se suele hacer para obtener un mejor control visual del estilo.

Controlar las opciones por defecto

Por defecto, la mayoría de los navegadores establecen que todos los enlaces estén subrayados y los enlaces en estado "seleccionado" tienen un contorno, como se puede ver en la figura 22:

Figura 22

De izquierda a derecha: los estilos por defecto de "seleccionado" de Opera 9, Firefox 2 e IE 7

Si queréis sustituir estos estilos por otros, podéis cambiar o desactivar estos valores predeterminados.

Subrayado

El subrayado (*underline*) se define mediante la propiedad `text-decoration`*:

* http://www.w3.org/TR/REC-CSS2/text.html#lining-striking-props

```
a {
  text-decoration: underline;
}
```

Podéis desactivar el subrayado estableciendo la propiedad en `none`:

```
a {
  text-decoration: none;
}
```

Incluso si mantenéis el estilo del subrayado, puede que os resulte más fácil desactivar `text-decoration` y utilizar `border-bottom` para establecer un subrayado falso. En el ejemplo que se muestra más adelante lo veremos más detenidamente.

Contorno

El contorno del estado "seleccionado" se controla con la propiedad `outline`*. El contorno es muy parecido a un borde, pero no ocupa espacio ni provoca que la página refluya cuando aparece (tened en cuenta que IE 7 y versiones anteriores no lo soportan). La manera más fácil de controlar el contorno es con la propiedad de abreviación:

* http://www.w3.org/TR/REC-CSS2/ui.html#dynamic-outlines

```
a:focus{
  outline: thick solid #000;
}
```

Este ejemplo se representaría más o menos como se ve en la figura 23:

Figura 23

Representación de ejemplo de un contorno negro y grueso

6.2.6. Ejemplo: recreación de las opciones por defecto de Netscape

Para dar un ejemplo fácil de estilo de los enlaces, recrearemos las opciones por defecto de Netscape, es decir, el azul, el morado y el rojo. Mantendremos el

subrayado, pero haremos que el estado activo se muestre en cursiva. Aumentaremos el tamaño del texto sólo para ver cómo queda y estableceremos un fondo blanco para la página.

```
body{
  background: #fff;
  color: #000;
  font-size: 2em;
}
a {
  text-decoration: underline;
}
a:link{
  color: #0000CC;
}
a:visited{
  color: #6D006D;
}
a:focus{
  color: #CC0000;
}
a:hover{
  color: #CC0000;
}

a:active{
  color: #CC0000;
  font-style: italic;
}
```

De esta manera, conseguiremos un resultado parecido al de la figura 24:

Figura 24

Recreación de las opciones por defecto de Netscape

Subrayados falsos con `border-bottom`

Muchos diseñadores han observado que el subrayado es un poco demasiado grueso y tapa las astas descendentes de las letras minúsculas, es decir, la línea del subrayado cruza la parte inferior de las letras “g”, “j”, “p”, e “y”, tal como se muestra la figura 25:

Figura 25

La línea del subrayado corta las astas descendentes de las minúsculas.

Pongamos por caso que la persona que se encarga del diseño de la página está de acuerdo y le gustaría que la línea del subrayado fuera más fina y que no tocara el texto. Para cumplir este requisito, que es muy habitual, utilizaremos un borde en lugar de un subrayado de manera que quede como la figura 26:

Figura 26

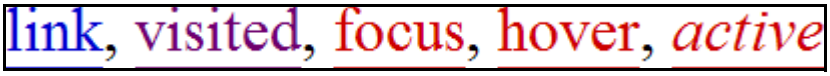
Utilizar un borde en vez del subrayado queda más bonito.

Primero, desactivad el subrayado para todos los estados de enlace y, después, estableced un borde inferior que encaje con el color del enlace en cada estado.

```
body{
  background: #fff;
  color: #000;
  font-size: 2em;
  line-height: 2em;
}
a {
  text-decoration: none;
}
a:link{
  color: #00c;
  border-bottom: 1px solid #00c;
}
a:visited{
  color: #6D006D;
  border-bottom: 1px solid #6D006D;
}
a:focus{
  color: #c00;
  border-bottom: 1px solid #c00;
}
a:hover{
  color: #c00;
  border-bottom: 1px solid #c00;
}
a:active{
  color: #c00;
  border-bottom: 1px solid #c00;
  font-style: italic;
}
```

De esta manera, conseguiremos un resultado parecido al de la figura 27:

Figura 27



Enlaces con el subrayado falso aplicado

Si utilizáis el método del borde falso, id con cuidado de tener suficiente `line-height` (altura de línea) establecido para que el subrayado no choque con el texto de la línea siguiente.

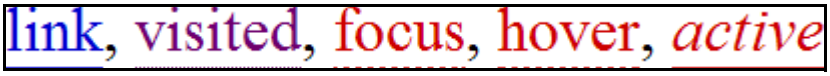
Estilos no basados en el color

Como el ejemplo que hemos utilizado hasta ahora se basa totalmente en los colores para distinguir cuatro de los cinco estados de los enlaces, deberíamos dar el paso siguiente y cambiar el borde inferior para los estados “visitado”, “seleccionado” y “por encima”. Demos a los enlaces visitados un borde de puntos y a los enlaces “por encima” y “seleccionado” un borde discontinuo:

```
body{
  background: #fff;
  color: #000;
  font-size: 2em;
}
a {
  text-decoration: none;
}
a:link{
  color: #00c;
  border-bottom: 1px solid #00c;
}
a:visited{
  color: #6D006D;
  border-bottom: 1px dotted #6D006D;
}
a:focus{
  color: #c00;
  border-bottom: 1px dashed #c00;
}
a:hover{
  color: #c00;
  border-bottom: 1px dashed #c00;
}
a:active{
  color: #c00;
  border-bottom: 1px solid #c00;
  font-style: italic;
}
```

De este modo, conseguiremos un resultado parecido al de la figura 28:

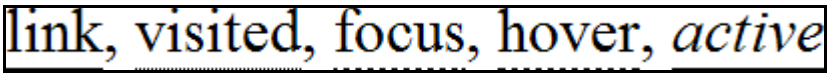
Figura 28



Cambiar el estilo de borde para cada estado de enlace

Este método, si se aceptan "focus" y "hover" como estados con estilos equivalentes, provoca que los estados de enlace se distingan con más factores además del color. Incluso si tuvierais que ver estos enlaces en blanco y negro, podríais identificar los diferentes estados de enlace tal como se muestra la figura 29:

Figura 29



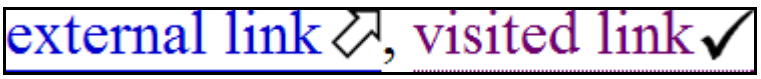
Ahora los estados de enlace pueden distinguirse aunque estén en blanco y negro.

6.2.7. Iconos en enlaces

Algunos sitios web utilizan iconos y símbolos para añadir información sobre los enlaces. Por ejemplo, algunas páginas utilizan una flecha para indicar que un enlace conduce a un sitio externo, o utilizan una marca de aprobado para indicar que el enlace ya se ha visitado.

Estos efectos son muy fáciles de aplicar con imágenes de fondo, como se muestra en la figura 30:

Figura 30



Ejemplo de enlaces con iconos distintivos

Para añadir un icono de flecha a los enlaces externos, podéis añadir la clase `external` a la etiqueta del enlace (`link`):

```
<a href="http://example.com/" class="external">external link</a>
```

Después, en vuestra hoja de estilo, aplicad una imagen de fondo para esta clase sin olvidaros añadir el relleno para que quepa la imagen:

```
a.external {
  background: #fff url("icon-external.gif") center right no-repeat;
  padding-right: 30px;
}
```

Este ejemplo aplicaría el icono a todos los casos de enlaces visitados, a todos los estados. Si quisierais limitar el icono sólo a los enlaces externos no visitados, podéis combinar las clases y las pseudoclases del estado de enlace en el selector.

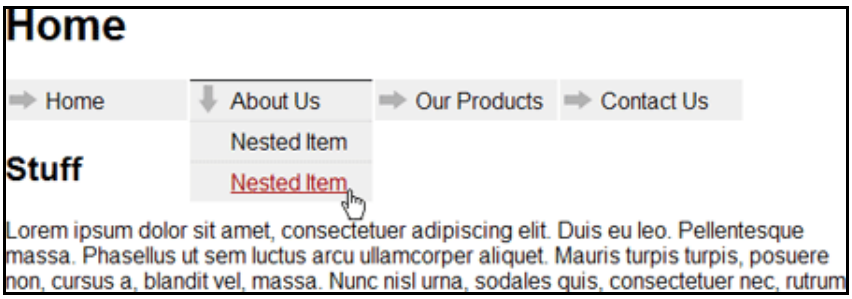
```
a.external:link{
  background: #fff url("icon-external.gif") center right no-repeat;
  padding-right: 30px;
}
```

Combinar clases y estados proporciona una amplia gama de posibilidades creativas para aplicar a los enlaces. Mientras recordéis comprobar los colores, el único límite a partir de aquí es vuestra imaginación.

6.3. Combinarlo todo: un sencillo menú de navegación

Para demostrar una manera de combinar las listas y los enlaces, el zip de ejemplos incluye un submenú desplegable de navegación sencillo, como se ve en la figura 31. Los submenús desplegables son un sistema de navegación muy habitual.

Figura 31



Captura de pantalla del submenú de ejemplo

Ved los ejemplos en: "Simple flyout navigation menu"
http://mosaic.uoc.edu/ac/le/operafiles/32/styling-lists-example-flyout.html

Resumen

Para un diseñador web es esencial saber muy bien cómo aplicar estilos a listas y enlaces porque se utilizan por todas partes. Se suelen combinar para crear la navegación por el sitio, y un estilo claro de los enlaces es vital para que un sitio web sea fácil de utilizar. Unos estilos de enlaces mal hechos pueden confundir mucho a todo el mundo e incluso pueden hacer que determinadas personas no puedan utilizar el sitio.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿Cómo se pueden elegir los estilos básicos de las listas, como por ejemplo picos cuadrados o números romanos, en una lista ordenada?

2. ¿Qué es un *sprite* de imagen y para qué se puede utilizar?

3. ¿Por qué el contraste de los colores es importante y cómo se puede garantizar que los colores de los enlaces sean de colores adecuados?

4. ¿Cuál es el orden correcto para aplicar estilos a cada estado de enlace?

Bibliografía

"Type and Colour" (un capítulo de *Building Accessible Websites*, de Joe Clark)
http://joeclark.org/book/sashay/serialization/Chapter09.html

"Juicy Studio: Highlighting Links"
http://juicystudio.com/article/highlighting-links.php

"Max Design-Simple, accessible external links"
http://www.maxdesign.com.au/presentation/external/

"Resource Center-Contrast Analyser 2.0 (Paciello Group)"
http://www.paciellogroup.com/resources/contrast-analyser.html

"A List Apart: CSS *Sprites* : Image Slicing's Kiss of Death"
http://www.alistapart.com/articles/*sprite*s

7. Estilos para tablas

Ben Buchanan

En ocasiones, parece que en el mundo actual del desarrollo web las tablas estén algo marginadas. Se hace tanto caso a la norma de "¡no utilicéis tablas!" que a veces la gente se olvida de que en realidad la frase es: "No utilicéis tablas para la composición". Las tablas van muy bien para su objetivo original, que es mostrar datos tabulares. Por ello es útil saber cómo aplicarles estilos correctamente.

Esta guía se centra en cómo aplicar CSS de manera eficaz para obtener estilos de tablas de datos claros y legibles. Además, también hablaremos de algunos requisitos habituales del diseño de tablas.

> **Nota**
>
> Puede que os resulte útil descargaros los ejemplos de código para las tablas que se muestran en este apartado desde el archivo "table-examples.zip", de manera que podáis ir siguiendo el apartado.

Descargad los ejemplos en: "table-examples.zip"
http://mosaic.uoc.edu/ac/le/operafiles/33/table-examples.zip

7.1. Estructura de las tablas

Antes de centrarnos en el CSS, fijémonos en los elementos estructurales básicos de las tablas a las que deberéis aplicar estilos:

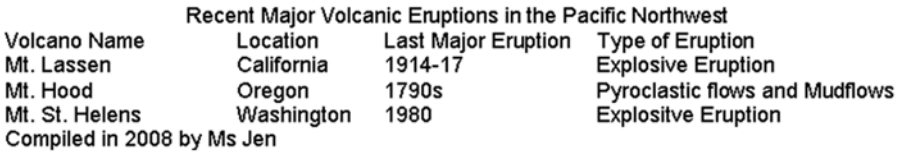
- Encabezamientos de las tablas.
- Celdas de datos de las tablas.
- Títulos de las tablas.

Cuando los usuarios de vuestro sitio lean vuestra tabla, deberían poder entender y seguir fácilmente su estructura. La manera más sencilla de conseguirlo es utilizar bordes, colores de fondo o ambos.

No es necesario que sigáis estas convenciones estilísticas, pero sí deberíais garantizar que existe una diferencia clara entre las celdas `th` y `td`; además, el `caption` (título) debería estar asociado claramente con la tabla y diferenciado del resto del texto de la página.

7.2. Conceptos básicos

Fijaos en cómo se representa esta tabla sin estilo:



Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explositve Eruption

Compiled in 2008 by Ms Jen

Podéis ver el mismo ejemplo en el apartado 5 del módulo "El texto central de HTML".

```
<caption>Recent Major Volcanic Eruptions in the Pacific Northwest</caption>
    <tr>
        <th>Volcano Name</th>
        <th>Location</th>
        <th>Last Major Eruption</th>
        <th>Type of Eruption</th>
    </tr>
    <tr>
        <td>Mt. Lassen</td>
        <td>California</td>
        <td>1914-17</td>
        <td>Explosive Eruption</td>
    </tr>
    <tr>
        <td>Mt. Hood</td>
        <td>Oregon</td>
        <td>1790s</td>
        <td>Pyroclastic flows and Mudflows</td>
    </tr>
    <tr>
        <td>Mt. St. Helens</td>
        <td>Washington</td>
        <td>1980</td>
        <td>Explosive Eruption</td>
    </tr>
</table>
```

Los datos son comprensibles, pero hay que esforzarse un poco para ver cómo va todo. Añadámosle unos estilos para hacer que sea más fácil de leer.

7.2.1. Anchura de tabla y de celda

La primera decisión que hay que tomar es qué anchura debe tener la tabla. La opción por defecto del navegador es la misma que establecer `table {width: auto; }`, que da como resultado que la tabla se extienda a lo ancho del contenido. Esto, en general, queda desordenado.

Pongamos por caso que nuestra tabla debe ir en una columna de contenido de 600 píxeles de anchura. Especificamos que la tabla se amplíe hasta el 100% de la anchura disponible para aprovechar mejor el espacio. Como hay cuatro columnas, establezcamos también la anchura de las celdas de tabla en un 25% cada una.

```
table {
  width: 100%;
}
th, td {
  width: 25%;
}
```

De hecho, podéis limitaros a establecer la anchura para `th` y esto definirá la anchura de todas las columnas; aun así, siempre va bien ser riguroso. Este estilo tan sencillo dará el resultado que se ve en la figura 1:

Figura 1

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption

Compiled in 2008 by Ms Jen

La tabla de ejemplo con una configuración sencilla de anchura

Ahora las celdas quedan con una anchura igualada. Más adelante ya veremos cómo especificar anchos desiguales, pero de momento seguimos con el siguiente paso.

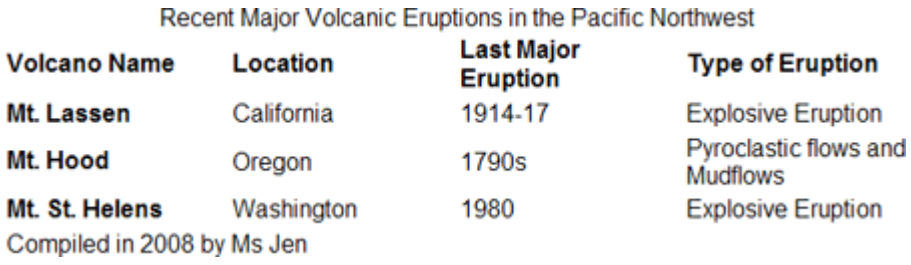
7.2.2. Alineación del texto

La tabla aún es un poco confusa de leer, de manera que vamos a especificar que la alineación del texto sea un poco más clara. La norma adicional que se muestra a continuación alineará a la izquierda los encabezamientos para que encajen con el contenido (por defecto, los navegadores centran los títulos de las tablas):

```
table {
  width: 100%;
}
th, td {
  width: 25%;
  text-align: left;
}
```

Así se clarifica un poco la tabla, como se puede ver en la figura 2:

Figura 2



Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption

Compiled in 2008 by Ms Jen

Tabla con alineación a la izquierda

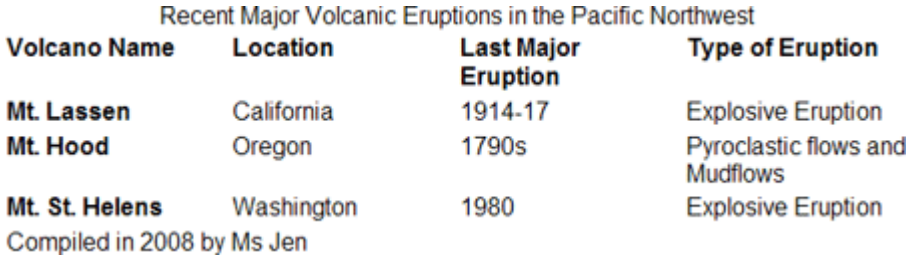
Ahora mismo, todas las celdas están alineadas verticalmente en el centro. Si lo preferís, podéis definir que el texto se alinee en la parte superior o inferior de la celda, o cualquier posición de `vertical-align*` que queráis. Las siguientes reglas establecen que el texto se alinee en la parte superior.

*** http://www.w3.org/TR/REC-CSS2/visudet.html#line-height**

```
table {
  width: 100%;
}
th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
}
```

Ahora la tabla queda como se ve en la figura 3:

Figura 3



Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption

Compiled in 2008 by Ms Jen

La tabla con alineación vertical

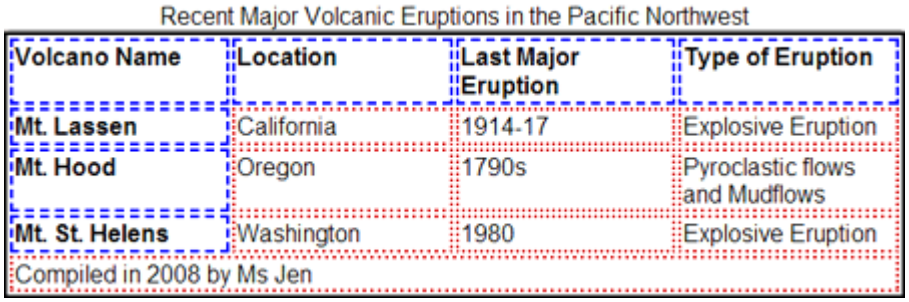
Fijaos en cómo todos los títulos de la fila superior quedan tocando arriba aunque “Last Major Eruption” queda en dos líneas.

7.2.3. Bordes

La tabla ya queda un poco mejor, pero aún cuesta un poco leer cada línea. Ahora aplicaremos algunos bordes para hacer que todo sea más fácil de leer. Debéis establecer los bordes de manera independiente para cada parte de la tabla y después decidir cómo se deberían combinar.

Para mostraros dónde quedarán los bordes, la figura 4 muestra diferentes bordes para `table` (`solid black`, negro sólido), `caption` (`solid grey`, gris sólido), `th` (`dashed blue`, línea discontinua azul) y `td` (`dotted red`, línea de puntos roja):

Figura 4



Muestra de los diferentes bordes en una tabla

Fijaos en cómo el borde de la `table` recorre el contorno exterior de todas las celdas de encabezamiento y de datos y después pasa entre las celdas y el título de la tabla. También podéis ver que, por defecto, los bordes `th` y `td` se distancian un poco el uno del otro.

Pasemos ahora a otro estilo de tabla. Podéis crear un borde negro sencillo para la tabla y las celdas mediante la propiedad `border` (borde). Esto se hace con las reglas siguientes:

```
table {
  width: 100%;
  border: 1px solid #000;
}
th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
}
```

y da como resultado lo que se muestra en la figura 4b:

Figura 4b

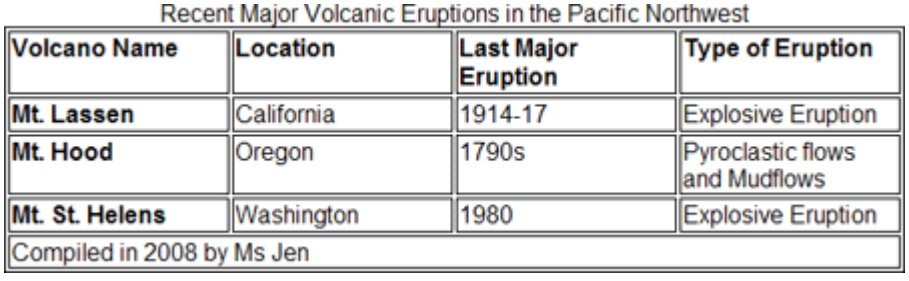


Tabla con bordes negros sencillos

Esto provoca que las filas sean más fáciles de leer, pero puede que no os guste el espacio que queda entre las celdas. Hay dos maneras de cambiarlo.

Primero, podéis limitaros a cerrar los huecos utilizando la propiedad `border-spacing` (espaciado de bordes), por ejemplo así:

```
table {
  width: 100%;
  border: 1px solid #000;
}
th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
  border-spacing: 0;
}
```

De esta manera, los bordes se tocarán en lugar de quedar separados. Así se cambia el borde de 1 píxel por un borde de 2 píxeles, como se ve en la figura 5:

Figura 5

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Tabla sin espaciado entre bordes que produce un efecto de borde de 2 píxeles.

También podéis aumentar el espacio entre las celdas utilizando `border-spacing`, aunque hay que tener en cuenta que esta propiedad no funciona con Internet Explorer.

Si queréis mantener el efecto de borde de 1 píxel, necesitaréis definir la tabla de manera que los bordes se "colapsen" los unos sobre los otros. Podéis conseguirlo utilizando la propiedad `border-collapse` (superposición de bordes) en lugar de la de `border-spacing` (espaciado de bordes):

```
table {
  width: 100%;
  border: 1px solid #000;
}
```

```
th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
  border-collapse: collapse;
  }
```

De esta manera, obtendréis una tabla con un borde de 1 píxel como se observa en la figura 6:

Figura 6

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Tabla con `border-collapse` (superposición de bordes) definido para superponerse y así reducir el borde en 1 píxel

Cuando defináis que los bordes se superpongan, debéis tener en cuenta que esto os puede causar problemas si tenéis estilos de bordes diferentes aplicados a celdas adyacentes. Cuando los estilos de bordes diferentes se superponen unos sobre otros, entrarán en "conflicto" entre ellos. Esto se soluciona mediante las reglas de resolución de conflictos de bordes de tabla de la especificación de CSS2 del W3C*, que determinan qué estilo "gana" cuando se superponen.

*** http://www.w3.org/TR/REC-CSS2/tables.html#border-conflict-resolution**

7.2.4. Relleno

Ahora que ya habéis aplicado bordes en las celdas, puede que queráis añadir un poco de espacio en blanco a las celdas del título y de la tabla. Para hacerlo, sólo debéis utilizar el `padding` (relleno).

```
table {
  width: 100%;
  border: 1px solid #000;
}
th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
  border-collapse: collapse;
  padding: 0.3em;
}
```

```
caption {
  padding: 0.3em;
}
```

Esto hace que el texto “respire” un poco, como se ve en la figura 7:

Figura 7

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Tabla con relleno aplicado a todas las celdas

7.2.5. Colocación del `caption`

Hasta ahora, el `caption` ha estado siempre en la parte superior de la tabla. Sin embargo, puede que queráis moverlo a otro lugar. Por desgracia, en IE no se puede hacer hasta la versión 8, pero en todos los demás navegadores podéis cambiar la posición del `caption` utilizando la propiedad `caption-side` (lado del título). Las opciones posibles son: `top` (superior), `bottom` (inferior), `left` (izquierda) y `right` (derecha). Ahora lo pondremos en la parte inferior:

```
table {
  width: 100%;
  border: 1px solid #000;
}
th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
  border-collapse: collapse;
  padding: 0.3em;
  caption-side: bottom;
}
caption {
  padding: 0.3em;
}
```

En la figura 8 se muestra el resultado:

Figura 8

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Recent Major Volcanic Eruptions in the Pacific Northwest

Tabla con el `caption` movido a la parte inferior

Para el resto de ejemplos, dejaremos el `caption` en la parte superior.

7.2.6. Fondo

Otra manera muy sencilla de aplicar estilos a las tablas es añadir colores e imágenes de fondo. Esto se hace con la propiedad `background` (fondo), aunque debéis tener presente que las diferentes partes de la tabla actuarán como "capas" una encima de la otra. La especificación CSS2 explica los efectos de capas de fondo bastante detalladamente*, pero en pocas palabras: los fondos se taparán los unos a los otros en este orden:

*** http://www.w3.org/TR/REC-CSS2/tables.html#table-layers**

1. tabla (que queda "al fondo del todo" o "detrás de todo"),
2. grupos de columnas,
3. columnas,
4. grupos de filas,
5. filas,
6. celdas (por "encima" o "delante" de todo, es decir, que su fondo tapa a todos los demás).

Por lo tanto, si establecéis un fondo para la tabla y un color diferente para las celdas, el fondo de la celda tapará el fondo de la tabla. Si habéis establecido los bordes en `collapse` (superposición), el fondo de la tabla no se verá. No obstante, si habéis establecido `border-collapse` en `separate`, el fondo de la tabla se verá entre los bordes.

Nota

Fijaos en que el concepto de elementos diferentes uno encima del otro en la página es controlable; podéis controlar a qué altura de la "pila" queda un elemento en relación con los demás cambiando su propiedad `z-index`.

En el apartado 11 de este módulo aprenderéis más cosas sobre `z-index`.

Imaginaos que definís la tabla de manera que tenga un fondo rojo y las celdas con fondo blanco. Entre la separación de las celdas se ve el rojo, pero las celdas siguen siendo blancas, como se muestra en la figura 9:

Figura 9

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Tabla que demuestra cómo el elemento de fondo de tabla rojo se ve entre los fondos blancos de las celdas.

También podéis utilizar una imagen de fondo. Por ejemplo, si queréis tener una gradación que se vea entre las celdas, podríais definir `th` y `td` con fondos blancos, pero establecer el fondo de `table` con una gradación de colores:

```
table {
  border-collapse: separate;
  border-spacing: 5px;
  background: #000 url("gradient.gif") bottom left repeat-x;
  color: #fff;
}
td, th {
  background: #fff;
  color: #000;
}
```

Fijaos en que el color de fondo se establece en negro, que llena el espacio en la parte superior donde acaba la gradación cromática (siempre debéis tener en cuenta que la tabla puede ser más alta que la imagen de fondo). El color de delante del todo se establece en blanco por si en algún momento estos colores por defecto se ven a través del contenido de las celdas. En general, los estilos de las celdas cancelarán la configuración del color del texto del estilo de `table {}`, pero siempre se deberían definir colores de fondo y de primer plano que hagan contraste.

Estos estilos crean una tabla que quedaría como en la figura 10 en la mayoría de los navegadores.

Figura 10

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Tabla con una imagen de fondo con gradación de colores que se ve entre las celdas

Por defecto, con las versiones anteriores a la 8 de IE no se verá tanto el fondo porque no reconoce `border-spacing`. Sin embargo, se consigue el mismo efecto, tal como se observa en la figura 11:

Figura 11

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

IE proporciona menos espacio entre bordes.

Dependiendo de las circunstancias, puede que os conforméis con esta diferencia de representación entre navegadores. Pero, lógicamente, eso no siempre es posible, por ejemplo cuando un cliente especifica que quiere que un diseño quede exactamente igual en todos los navegadores.

7.2.7. Arreglar IE con estilos condicionales

Existe una manera de solucionar los problemas de IE que hemos mencionado antes. Hay que hurgar un poco y requiere una hoja de estilo extra, pero funciona. Podéis utilizar una `expression` para hacer el espacio más ancho y después cargar esta expresión mediante comentarios condicionales. La sintaxis de la expresión es ésta:

```
table {
  border-collapse: expression("separate", cellSpacing = "5px");
}
```

Este CSS sólo sirve para IE, de manera que sólo éste lo debe poder aplicar. La expresión también invalidará la hoja de estilo, de manera que los desarrolla-

dores prefieren aislar las manipulaciones de IE en una hoja de estilo aparte que sólo cargue IE.

Para hacerlo, cread una hoja de estilo nuevo que se llame ie-only.css ("sólo IE") y enlazadla dentro de comentarios condicionales:

```
<!--[if lte IE 7]><link rel="stylesheet" media="screen" href="ie-only.css" />
<![endif]-->
```

Fijaos en que `[if lte IE 7]` significa "si menos que o igual a la versión 7 de IE". Esto revela el código en IE 7 y en todas las versiones anteriores de IE, pero el comentario de HTML que lo rodea oculta el código a todos los demás navegadores. Podéis ajustarlo a cualquier versión de IE que necesitéis, por ejemplo, si se trata de la versión IE6 y anteriores, utilizad: `[if lte IE 6]`.

En vuestra hoja de estilo principal, estableced el estilo normal:

```
table {
  border: 1px solid #000;
  border-collapse: separate;
  border-spacing: 5px;
  background: #000 url("gradient.gif") bottom left repeat-x;
}
```

y entonces estableced el estilo únicamente para IE como `ie-only.css`:

```
table {
  border-collapse: expression("separate", cellSpacing = "5px");
}
```

Así, IE mostrará una tabla con espacios anchos entre las celdas. Sólo debéis acordaros de conservar la configuración de anchura adicional. Si actualizáis vuestra hoja de estilo principal, también deberéis actualizar `ie-only.css`. Como es lógico, los comentarios condicionales permiten hacer muchas cosas más aparte de aplicar estilos a las tablas, porque la hoja de estilo extra puede incluir todo el CSS que necesitáis para arreglar errores del IE.

7.2.8. Un diseño sencillo

La mayoría de los diseños utilizan combinaciones sencillas de fondo. Por lo tanto, ahora aplicaremos a los encabezamientos de la tabla un fondo gris y cambiaremos el título de manera que sea texto blanco sobre fondo negro:

```
table {
  width: 100%;
  border: 1px solid #000;
}
th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
  border-collapse: collapse;
  padding: 0.3em;
  caption-side: bottom;
}
caption {
  padding: 0.3em;
  color: #fff;
  background: #000;
}
th {
  background: #eee;
}
```

Esto queda como se ve en la figura 12:

Figura 12

Recent Major Volcanic Eruptions in the Pacific Northwest			
Volcano Name	**Location**	**Last Major Eruption**	**Type of Eruption**
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Tabla con título de texto blanco sobre fondo negro y fondo gris claro en las celdas de encabezado

7.3. Variaciones habituales

En este subapartado nos centraremos en algunos arquetipos de diseño habituales que veréis muchas veces en muchas tablas de la web.

7.3.1. Aplicar líneas de cebra

Algo que se suele pedir mucho en relación con las tablas es crear filas de colores alternos. Se suelen denominar "líneas de cebra". Aunque existe cierta polémica sobre si las líneas de cebra ayudan o no realmente al lector*, son un estilo muy popular. En la figura 13 se muestra un ejemplo:

* http://www.alistapart.com/articles/zebrastripingdoesithelp

Figura 13

Common table elements		
Element	**Tag**	**Purpose**
table	<table>	Encloses a table
table row	<tr>	Encloses a row of table cells
table header cell	<th>	Sets a heading for a row or column
table data cell	<td>	Contains data
caption	<caption>	Sets a caption for the table

Tabla con "líneas de cebra", filas alternativas de color blanco o gris claro

La manera más fácil de aplicar el efecto de las líneas de cebra es añadir una clase para alternar filas de la tabla y después utilizar un selector de CSS contextual para aplicar estilos en las celdas de estas filas. Antes que nada, se añaden las clases "par" (*odd*) e "impar" (*even*) a las filas de la tabla, así:

```
 ...
<tr class="odd">
  ...
<tr class="even">
  ...
```

Podéis saltaros la fila del título porque ya tiene un estilo propio. Entonces añadid una clase contextual para definir el fondo de todas las celdas de las filas de la clase impar (*odd*).

```
.odd th, .odd td {
  background: #eee;
}
```

Ésta es la manera más sencilla de añadir líneas de cebra a una tabla de HTML que funcione en todos los navegadores, pero no es del todo perfecto porque, ¿qué sucede si después añadís una fila a la tabla? Pues que deberíais desplazar todos los nombres de clase `odd` y `even` para que todo quedara bien.

Hay dos opciones alternativas:

- Podéis añadir las clases mediante JavaScript no obstrusivo como se demuestra en *A List Apart: Zebra Tables**. La mayoría de los *frameworks* de JavaScript también tienen un método adecuado: *Zebra Table Showdown*** compara toda una gama de implementaciones.

- Podéis utilizar el selector de CSS3 `:nthchild`, pero aún no lo reconocen todos los navegadores. De todos modos, con el paso del tiempo eso irá mejorando.

* http://www.alistapart.com/articles/zebratables/

** http://jquery.com/blog/2006/10/18/zebra-table-showdown/

Podéis encontrar más información sobre las líneas de cebra con `nth-child` en un apartado de dev.opera.com*.

* http://dev.opera.com/articles/view/zebra-striping-tables-with-css3/

7.3.2. Parrillas incompletas

Algunos diseños responden bien con parrilas de aspecto más abierto y menos estructurado. Una variante muy sencilla consiste en eliminar los bordes verticales y dejar que el fondo llene el título, como en la figura 14:

Figura 14

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Tabla con bordes gris claro sólo por los extremos y por debajo de cada celda

El CSS para conseguir este efecto es:

```
table {
  width: 100%;
  border: 1px solid #999;
  text-align: left;
  border-collapse: collapse;
  margin: 0 0 1em 0;
  caption-side: top;
}
caption, td, th {
  padding: 0.3em;
}
th, td {
```

```
  border-bottom: 1px solid #999;
  width: 25%;
}
caption {
  font-weight: bold;
  font-style: italic;
}
```

También podéis ir un paso más allá y eliminar todos los bordes excepto el borde superior e inferior para dar un poco más de definición a la tabla central. Podéis ver la figura 15:

Figura 15

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption

Compiled in 2008 by Ms Jen

Tabla con bordes sólo en la parte superior e inferior de la tabla central

El CSS para conseguir este efecto es el siguiente:

```
table {
  width: 100%;
  text-align: left;
  border-collapse: collapse;
  margin: 0 0 1em 0;
  caption-side: top;
}
caption, td, th {
  padding: 0.3em;
}
tbody {
  border-top: 1px solid #000;
  border-bottom: 1px solid #000;
}
tbody th, tfoot th {
  border: 0;
}
th.name {
  width: 25%;
}
```

```
th.location {
  width: 20%;
}
th.lasteruption {
  width: 30%;
}
th.eruptiontype {
  width: 25%;
}
tfoot {
  text-align: center;
  color: #555;
  font-size: 0.8em;
}
```

7.3.3. Parrillas interiores

Puede que en ocasiones os interese eliminar el borde exterior pero conservar la parrilla interior de bordes como en la figura 16:

Figura 16

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption

Compiled in 2008 by Ms Jen

Tabla con un diseño de parrilla interior

Para aplicar este efecto a todos los navegadores actuales, debéis añadir una clase a las celdas `th` y `td` que aparecen en el último lugar de cada fila, así:

```
 ...
<tr>
  <th scope="col">Volcano Name</th>
  <th scope="col">Location</th>
  <th scope="col">Last Major Eruption</th>
  <th scope="col" class="last">Type of Eruption</th>
</tr>
  ...
```

Entonces utilizamos la clase para eliminar el borde derecho de estas celdas. El CSS completo es el siguiente:

```
table {
  width: 100%;
  text-align: left;
  border-collapse: collapse;
  margin: 0 0 1em 0;
  caption-side: top;
}
caption, td, th {
  padding: 0.3em;
}
th, td {
  border-bottom: 1px solid #000;
  border-right: 1px solid #000;
}
th.last, td.last {
  border-right: 0;
}
tfoot th, tfoot td {
  border-bottom: 0;
  text-align: center;
}
th {
  width: 25%;
}
```

Parrillas interiores con `:lastchild`

Cuando mejore la compatibilidad de los navegadores, podremos utilizar el pseudoselector `:lastchild` para conseguir este efecto sin clases. El CSS sería:

```
table {
  width: 100%;
  text-align: left;
  border-collapse: collapse;
  margin: 0 0 1em 0;
  caption-side: top;
}
caption, td, th {
  padding: 0.3em;
}
```

```
th, td {
  border-bottom: 1px solid #000;
  border-right: 1px solid #000;
}
th:lastchild, td:lastchild {
  border-right: 0;
}
th {
  width: 25%;
}
```

Esto ya funciona actualmente en las últimas versiones de todos los navegadores (el último navegador significativo que no lo admite es la versión 8 de IE).

Resumen

Ya deberíais dominar las opciones de estilo básicas para tablas. Hay ciertos límites que imponen las inconsistencias de los navegadores, pero en general deberíais poder crear tablas claras y legibles sin ningún problema. Sólo debéis fijaros bien en los bordes, dar al texto un poco de espacio para que pueda "respirar" e ir con cuidado con los fondos.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿Cómo se controla el espacio entre la tabla y los bordes de las celdas?

2. ¿Qué sucede cuando `table` (tabla) tiene un color de fondo, las celdas `th` y `td` tienen otro color de fondo y `border-collapse` (superposición de bordes) está establecido en `collapse` (superposición)?

3. ¿Cómo pueden establecerse anchos diferentes para cada columna?

Lecturas complementarias

W3C: CSS2 Tables, sobre todo el apartado que habla sobre las capas de fondo de tabla de CSS2.
http://www.w3.org/TR/REC-CSS2/tables.html

"A Dao of Web Design-dejad que la web sea la web". *A List Apart*. Un artículo clásico que explica por qué una diferencia de 1 píxel entre navegadores no es

realmente importante.
http://www.alistapart.com/articles/dao/

"Zebra Tables" y "Zebra striping: Does it Really Help?". En: *A List Apart*.
http://www.alistapart.com/articles/zebratables/
http://www.alistapart.com/articles/zebrastripingdoesithelp/

"Zebra striping tables with CSS3"
http://dev.opera.com/articles/view/zebra-striping-tables-with-css3/

"A CSS styled table" y "A CSS styled calendar". En: *Veerle's blog*
http://veerle.duoh.com/blog/comments/a_css_styled_table/
http://veerle.duoh.com/index.php/blog/comments/a_css_styled_calendar/

8. Diseño, composición y presentación de formularios con CSS

Ben Henick

El apartado de formularios HTML os presentó los conceptos básicos de creación de formularios y estilos. Este apartado continúa donde acabó el otro y ofrece más detalles sobre los elementos y estilos de formulario, así como sobre de qué modo se incluyen los formularios en los diseños de aplicación web del mundo real.

> **Nota**
>
> Podéis descargaros el fichero de código de ejemplo completo "styling_-forms_code.zip" para que podáis jugar con él en vuestro ordenador. Además, en varios puntos del apartado se ofrecen enlaces a los ficheros de ejemplo.

Descargad el fichero de código de ejemplo completo en: "Styling forms code examples"
http://mosaic.uoc.edu/ac/le/operafiles/34/styling_forms_code.zip

8.1. Conceptos nuevos que presenta este apartado

En este apartado encontraréis información sobre la implementación de formularios y la composición de interfaces.

8.1.1. Sobrecarga de reglas y etiquetas

Se puede decir que si se utilizan muchas etiquetas `class` e `id`, se incumple el principio KISS. Sin embargo, las composiciones exigentes a menudo provocan conflictos en la cascada, conflictos que se pueden resolver muy fácilmente añadiendo etiquetas y redactando reglas de hoja de estilo que contengan varios selectores.

El principio KISS se explica en el apartado 4 de este módulo.

Las composiciones más exigentes están llenas de casos límite, que es mejor solucionar asignando un `id` a los elementos que definen un contexto muy concreto y especial.

8.1.2. Elementos de campo de formulario nuevos

Un formulario efectivo a menudo debe ser algo más que un conjunto de botones y campos de introducción de texto porque es habitual estructurar las respuestas del usuario en términos de opciones. El HTML proporciona varias opciones para los diseñadores que se encuentren con este requisito.

8.1.3. Principios del diseño de formularios

Los formularios de un sitio web son normalmente puntos centrales para la interacción con el usuario y la obtención de datos. Por lo tanto, suelen ser fundamentales para el éxito de una web, que exige diseñarlos con todo el cuidado posible.

8.1.4. La regla de los tercios

Los usuarios suelen centrar su atención en cuatro puntos de un lienzo (y en las líneas que los atraviesan). Este apartado presenta este fenómeno al lector y ofrece sugerencias para aprovecharlo al máximo con el CSS.

8.1.5. Cuadrículas

En apartados anteriores ya se ha explicado cómo garantizar unas tipografías consistentes y sacar el máximo provecho del espacio en blanco. Este apartado va un poco más allá y explica cómo se pueden utilizar las unidades `em` para aplicar un grado de consistencia a la composición que sería imposible sin CSS.

Podéis ver cómo garantizar unas tipografías consistentes en el apartado 3 de este módulo y cómo sacar el máximo provecho del espacio en blanco en el apartado 5 del módulo "Conceptos de diseño web".

8.1.6. Capas de soporte de plataforma

Un requisito habitual de los proyectos comerciales es la representación casi exacta del diseño aprobado de la web en uno o más navegadores. En este apartado, se tratarán brevemente las causas, los efectos y los procesos relacionados en el cumplimiento de este requisito.

8.2. Un formulario de contacto sencillo

Los formularios de contacto con los que los visitantes de una web pueden enviar un correo electrónico directamente a una cuenta de correo son muy habituales por razones obvias: son accesibles a cualquier persona que tenga una dirección activa de correo electrónico y son fáciles de filtrar en una carpeta de correo específica.

Podéis ver los formularios de contacto en el apartado 6 del módulo "El texto central de HTML".

El etiquetado que se ha utilizado en un módulo anterior sobre formularios utiliza uno de estos formularios, que se ha decorado mucho.

Podéis ver el apartado 6 del módulo "El texto central de HTML".

8.2.1. Etiquetado

```
1  <form id="contactForm" method="post" action="/cgi-bin/service_email_script.php">
2    <ul>
```

```
3   <li id="nameField" class="required"><label for="realname">Name:</label>
       <input type="text" name="name" value="" class="medium" id="realname" />
        <span class="note">required</span></li>
4      <li id="addressField" class="required"><label for="address">Email:</label>
        <input type="text" name="email" value="" class="medium" id="address" />
        <span class="note">required</span></li>
5      <li id="subjectField"><label for="natureOfInquiry">General subject:</label>
6        <select name="subject" class="medium" id="natureOfInquiry">
7          <option value="support">Support</option>
8          <option value="billing">Accounts & billing</option>
9          <option value="press">Press</option>
10         <option value="other_q">Other questions</option>
11       </select>
12     </li>
13     <li id="acctTypeField"><label for="acctNone">Account type:</label>
14       <fieldset>
15         <label for="acctGold">Gold</label><input type="radio" name="acct_type"
           id="goldAcct" class="rInput" />
16         <label for="acctSilver">Silver</label><input type="radio"
           name="acct_type" id="acctSilver" class="rInput" />
17         <label for="acctBronze">Bronze</label><input type="radio"
           name="acct_type" id="acctBronze" class="rInput" />
18         <label for="acctNone">None</label><input type="radio" name="acct_type"
           id="acctNone" class="rInput" checked="checked" />
19       </fieldset>
20       <span class="note">required</span>
21     </li>
22     <li id="availabilityField">
23       <label for="availability">My account is unavailable:</label>
           <input type="checkbox"
           name="is_down" id="availability" class="rInput" /></li>
24     <li id="messageField"><label for="messageBody">Comments:</label>
         <textarea name="comments" cols="32" rows="8" class="long"
         id="messageBody"></textarea></li>
25     <li class="submitField"><input type="submit" value="Send"
        class="submitButton" /></li>
26   </ul>
27 </form>
```

8.2.2. Cambios con respecto al formulario anterior

Además de incluir varios elementos nuevos, se han añadido algunas clases e ID al etiquetado a las que se puede hacer referencia desde la hoja de estilo. De esta manera, se puede hacer referencia a cada formulario, pareja de campo/valor y campo de manera individual sea cual sea el contexto.

Los nuevos identificadores también hacen que sea más fácil que el diseñador pueda distinguir los campos que se deben llenar obligatoriamente de los que no.

Para terminar, hay algunas clases nuevas que ofrecen sugerencias de la cantidad y tipos de información que deberían mostrar los elementos de formulario donde están adjuntados. Estas clases permiten aplicar detalles de composición a elementos arbitrarios múltiples de manera simultánea.

8.2.3. Defectos aparentes

Como se supone que el formulario de demostración es el contenido principal, la `legend` (leyenda) utilizada en formularios anteriores se ha sustituido por un título.

Las leyendas son muy adecuadas en los `fieldsets` (conjuntos de campos) en vez de las `labels` o etiquetas (que pueden identificarse con controles únicos). En este caso, el elemento `legend` se ha omitido completamente porque es difícil aplicarle estilos.

Hay que destacar también que es mejor colocar las etiquetas "obligatorio" sobre los campos a los que les corresponda antes que el campo en el código fuente para facilitar las cosas a los usuarios con software de lector de pantalla. No obstante, la propiedad `position` (de la que no hablaremos en este apartado) es necesaria para disponer estos elementos de manera adecuada. Por lo tanto, las etiquetas "obligatorio" se han colocado *después de* sus controles asociados en el código fuente (aunque en el mismo contexto).

8.2.4. ¿Campos de formulario nuevos? ¿Qué es eso?

En el apartado anterior ya se habló de los campos de texto y de los controles de envío. Como ya se ha mencionado anteriormente, hay varios casos de uso que requieren que el usuario pueda seleccionar una de dos o tres opciones. Estos elementos se describen brevemente a continuación:

Elegir descripciones: `input type="checkbox"`

```
…
<label for="availability">My account is unavailable:</label>
   <input type="checkbox" name="is_down" id="availability" class="rInput" />
```

Las preguntas de participación o no participación suelen gestionarse con uno de estos controles. Otro caso en el que se utilizan es cuando hay que elegir ar-

bitrariamente entre varias opciones, como, por ejemplo, una lista de intereses personales.

Elegir entre estados mutuamente exclusivos: `input type="radio"`

```
          …
1         <label for="acctNone">Account type:</label>
2         <fieldset>
3           <label for="acctGold">Gold</label><input type="radio" name="acct_type"
            id="goldAcct" class="rInput" />
4           <label for="acctSilver">Silver</label><input type="radio" name="acct_type"
            id="acctSilver" class="rInput" />
5           <label for="acctBronze">Bronze</label><input type="radio" name="acct_type"
            id="acctBronze" class="rInput" />
6           <label for="acctNone">None</label><input type="radio" name="acct_type"
            id="acctNone" class="rInput" checked="checked" />
7         </fieldset>
```

Un grupo de éstos permite presentar varias opciones la una al lado de la otra y de las que sólo se puede elegir una. Un buen ejemplo de caso de uso de un conjunto de controles de `radio` (elección de un elemento entre varios) es la asignación de una nota en una escala de 1 a 5 o de 1 a 10.

Al contrario que otros controles de formulario, los controles de `radio` no sólo permiten sino que, de hecho, requieren que los controles asociados tengan el mismo `name` (nombre).

Estos elementos toman su nombre de una interfaz habitual en los aparatos de radio de coche sintonizados mecánicamente. Al contrario que los canales programables típicos de los aparatos de sintonización digital, los botones mecánicos de memoria, cuando se pulsan, suelen centrar el receptor en una gama de frecuencias de la banda que se está sintonizando.

Los controles de `checkbox` (casilla de selección) y `radio` permiten un atributo `checked` (comprobado) que, si se aplica, activa el control por defecto la primera vez que se representa.

Antes de responder a la pregunta de si utilizar controles de `radio` en lugar de controles de `checkbox`, hay que tener en cuenta varios factores. Si queréis que el usuario confirme una decisión subjetiva (como por ejemplo ser miembro de una lista de correo), probablemente serán mejores los controles `checkbox`. Si preferís que el usuario elija entre dos opciones objetivas (como, pongamos por caso, el género), es mejor utilizar los controles de `radio`.

Cuando hay demasiadas opciones para elegir: `select/option`

```
…
 <label for="natureOfInquiry">General subject:</label>
   <select name="subject" class="medium" id="natureOfInquiry">
     <option value="support">Support</option>
     <option value="billing">Accounts & billing</option>
     <option value="press">Press</option>
     <option value="other_q">Other questions</option>
   </select>
```

Los elementos `select` y `option` ofrecen resultados parecidos a los proporcionados por una serie de controles de `radio`, pero ocupan mucho menos espacio. Elegir un elemento `select` en lugar de una serie de controles de `radio` suele ser una cuestión de cómo se utiliza el espacio en la interfaz de usuario; una lista larga de zonas geográficas o departamentos en una web de comercio electrónico suele ser más adecuada para elementos `select`, mientras que una serie más corta de opciones (como sí/no, verdad/mentira, gama de edad, gama de ingresos) debería establecerse como una serie de controles de `radio`.

Si ponemos a prueba de manera meticulosa nuestra creación, comprobaremos que el nivel de control motriz necesario para manipular una lista `select` es elevado, pero aumenta ligeramente en proporción con el número de `options` que incluye. El resultado práctico es que a las listas cortas de opciones mutuamente exclusivas es mejor aplicarles un formato como una serie de controles `radio` con `labels` bien escritas.

Agrupar series de controles: `fieldset`

La función principal del elemento `fieldset` es asignar un único contexto a una serie de controles íntimamente relacionados (controles `text` para un número de teléfono, elementos `select` para una fecha, etc.).

8.3. De cero hasta un formulario completo

Ahora que ya hemos presentado brevemente el material nuevo de este apartado, ha llegado el momento de ver este material en acción: las doce demostraciones que siguen progresan a través de varios conceptos de diseño y retos de aplicar estilos que se pueden encontrar a la hora de crear formularios web.

Se recomienda a los lectores que guarden el material de demostración en su disco duro y que hagan pruebas con las reglas de hoja de estilo.

Nota

Estas demostraciones avanzan por orden de código fuente en lugar de por el orden de creación de la hoja de estilo. Más adelante en este apartado se habla de los motivos y las implicaciones de esta divergencia.

8.3.1. Demostración 1

Si empezamos con una regla como: `html {margin: 0; padding: 0; }`, el primer paso es configurar el `body` (texto central) de la página.

```
body{
  margin: 0;
  padding: 1.714em;
  background-image: url(images/bg_grid.gif);
  font-size: 14px;
  font-family: Helvetica,Arial,sans-serif;
  line-height: 1.714em;
}
```

Archivo fuente de: "Página sin demasiados estilos"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form_unstyled.html

Archivo fuente de: "Página con estilos `body` aplicados"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form01.html

8.3.2. Demostración 1: consideraciones previas

- Cuando el XHTML se sirve con el `Content-Type` adecuado, para un agente de usuario que lo acepte como es debido, el `margin` (margen) y/o `padding` (relleno) de la página por defecto se representan en el elemento `html`.

- En el resto de casos no descritos en el párrafo anterior, se coloca un canal de 10 píxeles en torno al perímetro de la página; Opera ofrece este elemento como valor de `padding`, pero otros navegadores populares lo ofrecen (de una manera un poco contraintuitiva) como valor `margin`. La hoja de estilo de demostración normaliza el resultado.

- Aunque muchos puristas de la accesibilidad se oponen al valor de tamaño de letra de 14 píxeles, es integral de las propiedades de caja y tipografía que se han especificado en otra parte de la hoja de estilo, básicamente denominados en

séptimas partes de un `em`. Al final del apartado se proporciona una tabla de conversión de fracciones a decimales para aquellos que quieran saber cómo es la aritmética utilizada.

- Se ha elegido el valor `14 px` porque es el tamaño más pequeño de texto central que puede leer casi todo el mundo con la vista corregida.
- Ya que uno de los objetivos de este apartado es demostrar el trabajo que se debe hacer para obtener una cuadrícula extremadamente consistente, se ha aplicado a la página un fondo de cuadrícula en incrementos de 24 píxeles.

8.3.3. Demostración 2

Ahora que hemos hablado de los contenedores de la página, los próximos dos pasos modifican o eliminan los estilos de agente de usuario.

```
h3 {
  margin: 0 0 1.2em 0;
  border-bottom: .05em solid rgb(0,96,192);
  font-size: 1.429em;
  line-height: 1.15em;
}
form {
  width: 35.929em;
  margin: 0;
}
ul {
  margin: 0;
  padding: 0;
  list-style-type: none;
}
```

Archivo fuente de: "Aplicar estilo al título principal y eliminar los canales no deseados"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form02.html

8.3.4. Demostración 2: consideraciones previas

- La progresión del tamaño de la letra de los títulos es diferente en cada plataforma; sin embargo, los valores predeterminados siempre son una proporción del valor `medium` utilizado para el texto de párrafo sin estilo y, por lo tanto, se heredan a través de la cascada. El resultado del valor proporcionado debe cambiar la proporción predeterminada.
- Se considera óptimo utilizar `h1` para el primer título de una página. En este caso, esta práctica se ignora porque, en un entorno de producción comercial, el título del sitio generalmente se encuentra en un `h1` de la página y el título de la página debería estar más abajo en la jerarquía de encabeza-

mientos. En muchos casos, la prominencia del formulario será igual a la prominencia de otros contenidos o formularios del mismo documento.

- El objetivo de los estilos de `h3` es crear una caja de contenido de 24 píxeles de altura con 24 píxeles de canal justo debajo, de manera que:

```
(((14 × 1.429) × 1.15) + (20 × .05)) ≈ 24
14 * 1.429 ≈ 20; 20 * 1.15 == 23; 20 * .05 == 1;

(20 × 1.2) = 24
```

- La asignación de un valor `width` (anchura) tanto en el `form` (formulario) como en los objetos de la lista es necesaria si los elementos se deben justificar adecuadamente sin confiar en el posicionamiento. El valor utilizado produce un valor estático de 503 píxeles; la única discrepancia de píxeles (teniendo en cuenta una unidad de cuadrícula atómica de 24 píxeles) es un producto de errores producidos por el redondeo y el *antialiasing*.

- Los estilos predeterminados de agente de usuario para listas varían de un navegador a otro. Internet Explorer da a cada elemento de la lista un margen izquierdo de 40 píxeles y coloca el símbolo de entrada de lista en el canal creado, mientras que otros navegadores aplican relleno en la parte izquierda del bloque de contenido del formulario en general. De la misma manera que las propiedades de composición modificadas en la regla `body`, esta regla está pensada específicamente para normalizar la presentación en todos los navegadores.

8.3.5. Demostración 3

Pasemos ahora a establecer los contenedores de los elementos del formulario.

```
li {
  clear: both;
  height: 1.714em;
  margin: 0;
}
fieldset {
  height: 1.429em;
  margin: 0 0 -.143em 0;
  border: 0;
}
```

Archivo fuente de: "Elementos de la lista de estilo (contenedores de la pareja de etiqueta/valor) y del `fieldset`"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form03.html

8.3.6. Demostración 3: consideraciones previas

- Si consideramos el formulario como una serie de *filas*, veremos que es necesario aplicar estilos a la altura de cada uno para preservar una cuadrícula. Los márgenes de los elementos de la lista se establecen a cero de cara a Internet Explorer y por si acaso de cara al resto.

- Como a muchos de los elementos del formulario se les aplicarán valores `float`, a los elementos de la lista que incluye se les aplica un valor de `clear: both` para garantizar que cada uno quedará por debajo de su predecesor automáticamente.

- Aparte de la eliminación del margen (que es parte del estilo por defecto del agente de usuario), las propiedades de composición del `fieldset` parecen haberse establecido de manera arbitraria. De hecho, se las asignó después de pruebas en todos los navegadores, de las que hablaremos de nuevo brevemente en las notas adjuntas a la Demostración 11.

- Llegados a este punto, no se han asignado valores `display` ni `float`, lo que explica por qué el contenido del `fieldset` choca con el siguiente control `select`.

8.3.7. Crear una cuadrícula

Uno de los puntos más significantes del buen diseño gráfico (y, de paso, del buen diseño de interfaces) es que los elementos se disponen a intervalos regulares. A estos intervalos se les suele denominar la **cuadrícula** (o *grid*, en inglés).

Como ya se ha mencionado anteriormente, la unidad atómica de cuadrícula del formulario de demostración son 24 píxeles cuadrados, pero la composición coherente va más allá de garantizar que los elementos de diseño se sitúen a intervalos pequeños pero regulares. Una cuadrícula realmente efectiva tiene las características siguientes:

- Los márgenes de columna se sitúan a intervalos consistentes de la cuadrícula en todo el documento.

- Las secciones secuenciales de un mismo documento comparten sus márgenes superiores con los elementos de columnas adyacentes.

- Las ilustraciones de una composición se recortan o se enmascaran hasta que tienen unas dimensiones que respetan las anchuras tanto de las columnas como de los intervalos de cuadrícula atómica.

- Incluso en los casos en los que el contenido está en una sola columna, puntuado por unos cuantos elementos `float`, todos estos últimos elementos tienen los mismos estándares de tamaño y composición.

Las composiciones que manifiestan estas características son más atractivas y fáciles de seguir, lo que contribuye a que el sitio sea más usable.

Crear la estructura de una cuadrícula en una composición

La herramienta que utilizan la mayoría de los profesionales para crear composiciones para la web es Adobe Photoshop, y una de las ventajas que tiene es el fácil acceso a las líneas de cuadrícula que ofrece. Para visualizar una cuadrícula atómica con Photoshop, se puede seleccionar View (vista) → Show (mostrar) → Grid (cuadrícula), con lo que se mostrarán las líneas de la cuadrícula en los intervalos establecidos en Guides & Grid Preferences (guías y preferencias de cuadrícula).

Superponer guías para elementos como las columnas se consigue seleccionando View (vista) → Rulers (reglas), cambiando a la herramienta Move (mover) y arrastrando el puntero desde la regla como sea necesario.

Aplicar la cuadrícula en la hoja de estilo

Como se comenta en el apartado de los estilos de texto, y refuerzan algunas de las reglas que se incluyen en la hoja de estilo de demostración, la mejor manera de aplicar una cuadrícula atómica en una composición es la de confiar en las unidades em. Sin embargo, sólo con eso no es suficiente; el especialista en estilos también debe mantener la corrección de las conversiones de fracciones a decimales cuando trata con otros tamaños de tipografía, canales y márgenes.

En la hoja de estilo de la demostración se puede ver otra técnica para aplicar cuadrículas: la provisión de etiquetas `class` (clase) que pueden hacer referencia a varios tamaños de elementos y columnas de un documento. Cuando se utiliza de manera consistente, estas etiquetas se encargan de casi todo el proceso de aplicación de la cuadrícula.

8.3.8. Demostración 4

Mantener los elementos alineados en una cuadrícula significa asignar propiedades de composición en las etiquetas y controles de formulario.

```
label {
  display: block;
  float: left;
  clear: left;
  width: 10.286em;
  overflow: auto;
  padding-right: 1.714em;
```

```
  text-align: right;
}
input {
  height: 1.143em;
  border: .071em solid rgb(96,96,96);
  padding: .071em;
  line-height: 1;
}
```

Archivo fuente de: "Alinear las dos columnas principales"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form04.html

8.3.9. Demostración 4: consideraciones previas

- *Todos* los controles de formulario, incluidas las áreas de texto y las etiquetas, se representan como elementos en línea por defecto.

- Para crear una columna izquierda consistente hay que asignar una `width` a los elementos `label`; en el modo "estricto" de representación, el `padding` proporcionado posibilita un canal factible entre controles y etiquetas.

- Hacer que tanto las etiquetas como los controles queden a la misma altura contra el mismo margen hace que el formulario sea fácil de leer. En el tema de la regla de los tercios se trata este hecho y otros aspectos de la composición (podéis verlo a continuación).

- Llegados a este punto, hay problemas claros con el formulario:

  – A las `labels` adjuntadas a los controles `radio` se les asignan los mismos valores de `display` y `float` que a las demás `labels` del formulario. En algunos navegadores, junto con los valores existentes de `height` y `overflow`, estos elementos chocan con la siguiente pareja de control y etiqueta.

  – En Safari 3, los márgenes de los controles de texto desaparecen en este desplazamiento. Se sospecha que es un error de representación.

  – Los controles `radio` parecen estar tocándose sin separación en el orden del código fuente; esto sucede porque los controles `label` que intervienen están en un contexto de composición diferente.

  – Otra curiosidad que vale la pena mencionar es la asignación de `overflow: auto` en etiquetas. El truco de magia que se aplica en este caso puede describirse como una regla: si tenemos un elemento de bloque dentro de otro, y teniendo en cuenta que sólo uno de ellos tiene una

`height` especificada en unidades estáticas o ems que se extienden a un número determinado de píxeles, asignar `overflow: auto` al otro elemento (lo que no tiene un valor de `height`) hará que se extienda hasta adquirir la altura del elemento que sí tiene un valor de `height` diferenciado. Esta técnica también se utiliza en la demostración 11.

8.3.10. La regla de los tercios

Figura 1

Una escena de principios de primavera en Portland, Oregon. Sobre esta foto se han superpuesto unas líneas para demostrar la regla de los tercios; fijaos en que la intersección inferior derecha y las líneas que la forman contienen toda la actividad visible. Foto ©2000 del autor; todos los derechos reservados.

Si se quiere conseguir una buena composición, hay un recurso casi universal: si dividís en tercios una disposición o ilustración, descubriréis que el observador centra la mirada en las líneas (y sobre todo en las intersecciones de las líneas) que señalan estas divisiones. Si no se aprovecha esta extraña tendencia de la vista, la composición quedará desequilibrada.

La explicación más sencilla de este fenómeno es que estas cuatro líneas se adaptan prácticamente a la perfección a la cuadrícula que sigue la sección áurea, una proporción próxima a un sexto. La sección áurea suele encontrarse en diferentes campos de las matemáticas y en el mundo natural.

Figura 2

Una captura de pantalla de msnbc.msn.com con los primeros siete rectángulos áureos superpuestos. Las dimensiones del cuarto y quinto rectángulo uno al lado del otro aclaran el carácter de la cuadrícula de composición de página en general.

El formulario de demostración se ha dispuesto de manera que los controles del formulario queden justificados en el margen izquierdo situado a un tercio de la distancia del margen derecho del formulario en conjunto; fue una decisión tomada a propósito. Mucho menos a propósito es la composición vertical del formulario porque, cuando se tiene en cuenta el título, los campos de texto cortan las dos líneas descritas en el párrafo anterior. Y aunque no se tenga en cuenta el título, los campos obligatorios cortan la parte superior de estas líneas.

El punto principal para un especialista en estilos es que si la sección áurea y la cuadrícula se tienen en cuenta antes de empezar a crear una hoja de estilo, el hecho de normalizar la hoja de estilo se puede simplificar de manera considerable.

8.3.11. Demostración 5

Para garantizar que la cuadrícula que queremos crear se conserve vertical y horizontalmente, se deben tener en cuenta algunos detalles. Estos cambios son casi del todo estéticos.

```
textarea {
  height: 4.714em;
  margin-bottom: .286em;
  border: .071em solid rgb(96,96,96);
  padding: 0;
}
select {
  display: block;
  float: left;
  height: 1.571em;
  font-family: Futura,'Century Gothic',sans-serif;
}
option { font-size: 100%; }
```

Archivo fuente de: "Retocar la presentación de los controles `textarea` y `select`"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form05.html

8.3.12. Demostración 5: consideraciones previas

- En las plataformas Windows, los controles `select` pueden modificarse para eliminar parte del biselado, lo que hace que se parezcan más a los controles de texto.

- Como la facilidad de uso suele mejorar dándole al usuario una manera de distinguir a primera vista los datos que introduce del resto del formulario, la tipografía utilizada para representar los valores del formulario será dife-

rente de la utilizada en el resto del formulario. Teniendo eso en cuenta, esta demostración aplica el primero de estos valores.

- La cascada no suele tenerse en cuenta en cuanto a la representación del texto en los controles del formulario, lo que es otra explicación de por qué hay varios valores de texto/fuente. Se ha evitado `inherit` (heredar) por razones de preferencia y costumbre y no por ningún motivo objetivo.

8.3.13. Demostración 6

La demostración previa manipula algunas representaciones de tipografía, de modo que ahora es el momento de acabar de redondear el trabajo.

```
input, textarea {
  display: block;
  float: left;
  overflow: hidden;
  font-family: Futura,'Century Gothic',sans-serif;
  font-size: 1em;
}
input, textarea, select {
  margin-top: 0;
  font-size: 100%;
}
```

Archivo fuente de: "Normalizar la presentación de los controles de texto y ajustar el efecto de la cascada en el texto de control de `select`"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form06.html

8.3.14. Demostración 6: consideraciones previas

- En esta demostración vemos el primer ejemplo de valores que se han reservado a propósito para una asignación simultánea en selectores múltiples. La ausencia de valores `border` de estas reglas se debe al proceso de producción, que se realizó en un orden diferente al de la presentación de estas demostraciones, un punto del que se habla más detalladamente más adelante.

- Como hemos mencionado anteriormente, los valores de texto y fuente de los controles de **formulario** no se benefician de las ventajas de la cascada. Estas reglas solucionan este hecho. Por lo tanto, la mayoría de los diferentes controles ahora ya se adaptan a la composición deseada.

- A causa del comportamiento de Internet Explorer 6, en el balance de los controles del formulario se les da un valor `float` de `left`. Este valor se

asignó por costumbre y por ciertas experiencias (desagradables), pero no es estrictamente necesario.

- Al asignar valores `block` a los controles de texto, el problema de representación de Safari visto en las dos demostraciones anteriores ya se ha solucionado. Las cosas raras como éstas son habituales cuando se aplican estilos a formularios.

- De la misma manera que los valores `border` no se han normalizado como es debido, tampoco se han normalizado los valores `font-size`; la asignación de valores `1 em` a los controles de texto y valores `100%` a los controles `select` fue totalmente arbitraria.

8.3.15. Demostración 7

Las anchuras de los distintos controles de texto se deben cambiar desde sus valores por defecto.

```
.medium { width: 11.714em; }
select.medium { width: 12em; }
.long { width: 20.429em; }
.rInput { border: 0; }
```

Archivo fuente de: "Modificar las anchuras de los controles de texto para hacerlas más usables o, como mínimo, más consistentes"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form07.html

8.3.16. Demostración 7: consideraciones previas

- Una reexaminación del etiquetado de origen revela que a cada control se le ha asignado una `class`: tres relacionadas con la anchura para el texto y los controles `select` y otras para los controles que se basan en información del puntero/cursor y no en información tecleada.

- Las clases se deben asignar a controles básicamente porque el soporte de Internet Explorer para los selectores avanzados es limitado. En cuanto a selectores, la clase `rInput` podría sustituirse fácilmente por `input[type="radio"]`, `input[type="checkbox"]`, etcétera. si las versiones de producción actuales de Internet Explorer aceptaran los segundos.

- La asignación de *tres* posibles valores para controles de texto es totalmente arbitraria y se deja a la voluntad del diseñador. En los entornos comerciales, algunos diseñadores entregan unos trabajos tan particulares en lo que respecta a la composición, que los selectores de `class` como los que se utilizan aquí serían funcionalmente inútiles. Asignar un `id` a cada pareja de

etiqueta/control ofrece la referencia más sencilla posible para cada elemento del formulario, una sencillez que es valiosa a la hora de aplicar estilos al trabajo de un diseñador que insiste en dar su toque a cada pequeño detalle de la composición del sitio.

8.3.17. Demostración 8

El botón "enviar" del formulario decae...

```
.submitButton {
  display: block;
  clear: both;
  width: 7.2em;
  height: 2em;
  margin: 0 0 0 16.8em;
  border: 1px solid rgb(128,128,128);
  padding: 0;
  font-size: 10px;
  text-align: center;
}
```

Archivo fuente de: "Ajustar con precisión la composición del botón 'enviar' del formulario"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form08.html

8.3.18. Demostración 8: consideraciones previas

- El reto más complicado al que nos enfrentamos cuando aplicamos estilos a botones de envío es el hecho de situarlos exactamente donde el diseñador quiere. En la práctica, el aspecto deseado se obtiene después de muchos ajustes de la composición y las propiedades `line-height`; algunos desarrolladores puede que encuentren menos laborioso utilizar una sustitución de imagen (podéis ver la bibliografía) o `input type="image"`.

- A primera vista, la asignación de `display: block` para este elemento parece redundante y, de hecho, lo es si pensamos en un único formulario de una única página. Sin embargo, en el contexto de todo un sitio web, podría no ser redundante; muchos sitios web y aplicaciones tienen múltiples formularios en un documento con diferentes valores de `display`.

8.3.19. Demostración 9

Ponemos las etiquetas "required" (necesario) donde correspondan.

```
li.required span.note {
  display: block;
```

```
  width: auto;
  float: right;
  color: rgb(128,128,128);
  font-size: .714em;
  line-height: 2.4em;
  font-style: italic;
}
```

Archivo fuente de: "Mover las etiquetas 'required' de manera que queden tocando en el margen derecho teórico del formulario y cambiar sus propiedades de texto"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form09.html

8.3.20. Demostración 9: consideraciones previas

- En su estado actual, el `fieldset` que contiene los controles `radio` sigue siendo un elemento de bloque, de manera que se extiende hasta el margen derecho del formulario. Por lo tanto, la etiqueta asociada a este conjunto de controles se desplaza por debajo del final calculado del `fieldset`.

- El valor `auto` que se proporciona para la `width` de las etiquetas "required" dicta que no deberían ser más anchos que su contenido.

- Si nos fijamos más en la aritmética utilizada para la tipografía de las etiquetas "required", veremos un tamaño de letra de 10 píxeles y un espacio interlineal de 24 (equivalente a una unidad atómica de la cuadrícula que se utiliza).

- La estructura utilizada para indicar los campos obligatorios se ha creado pensando en la interactividad con el usuario; con las distintas clases aplicadas al formulario es posible validar los datos introducidos por el usuario con JavaScript y cambiar los estilos de las etiquetas, campos y/o etiquetas en los conjuntos etiqueta/control que el usuario no ha manipulado adecuadamente.

8.3.21. Demostración 10

Finalmente, ha llegado el momento de establecer las colisiones de los controles `radio` con los campos que hay debajo en el orden de las fuentes.

```
fieldset label {
  margin-right: .25em;
  padding-right: 0;
  line-height: 1;
}
fieldset .rInput { margin-right: .75em; }
```

```
fieldset label, fieldset .rInput {
  width: auto;
  display: inline;
  float: none;
  font-size: .857em;
}
li.required fieldset {
  width: 18.857em;
  float: left;
}
```

Archivo fuente de: "Alinear los controles radio y sus etiquetas horizontalmente"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form10.html

8.3.22. Demostración 10: consideraciones previas

- El resultado principal de estas reglas, aparte de hacer los ajustes estéticos, es cambiar el valor `display` de los controles `radio` y `checkbox` de nuevo a `inline`. Esto se hace para evitar los problemas que supone aplicarles `float` como al resto de los elementos `input` del formulario.

- A pesar de la asignación de `display: inline` a los controles asociados, se quedan como elementos "sustituidos": elementos en línea con dimensiones estáticas conocidas en tiempo de ejecución (es decir, antes de que el navegador empiece a representar el contenido). Por este motivo pueden aplicarse márgenes a estos elementos.

- El carácter peculiar de los elementos `fieldset` (es decir, que son los únicos elementos de bloque pensados específicamente para utilizarse en formularios) hace que en este caso sea necesario aplicar una anchura diferenciada a cualquier control que incluya `fieldset` que necesite activación por parte del usuario. (Podéis ver los párrafos anteriores sobre la composición de etiquetas "necesarias".)

8.3.23. Demostración 11

Finalmente, para acabar como unos señores y conseguir que los últimos detalles queden alineados como es debido...

```
#acctTypeField fieldset {
  padding: .286em 0 0 0;
  line-height: normal;
}
```

```
#acctTypeField .rInput { margin-top: .167em; }
#availabilityField label {
  height: 3.143em;
  padding-top: .286em;
  line-height: normal;
}
#availabilityField .rInput { margin-top: .286em; }
#availabilityField, #messageField {
  height: 1%;
  overflow: auto;
}
```

Archivo fuente de: "Retocar los últimos detalles en varios controles"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form11.html

8.3.24. Demostración 11: consideraciones previas

- La magia del `overflow` que hemos aplicado en la demostración 4 se repite en este caso; `#availabilityField` tiene una `label` de `height` conocida y sucede lo mismo con la `textarea` de `#messageField`.

- El uso de la etiqueta `#acctTypeField` para cambiar el valor `padding` del `fieldset` que incluye puede ser demasiado específico. No obstante, hay que trabajar de manera precisa a la hora de redactar estilos que pueden verse afectados tan fácilmente por los estilos de los elementos adyacentes.

- El resto de las reglas de este bloque de la hoja de estilo acaban de retocar un poco la composición; se trata de retoques que se determinaron durante el proceso de pruebas. Por desgracia, las horas de comprobación y retoques no consiguen revelar el comportamiento que producirá controles `radio` de idéntica composición tanto en Safari 3 como en Firefox 3. El resultado es un punto de referencia en las `labels` de control de `radio` que es muy poco convencional en Firefox 3. En general, se puede decir que aplicar estilos a los controles `checkbox` y `radio` es una especie de magia negra.

8.3.25. Demostración 12

Todos los estilos anteriores se han desarrollado para Opera o Safari (elegid el que queráis, ambos se comportaron bastante bien). Los que se mencionan a continuación son específicos de Internet Explorer, especificados en un fichero CSS enlazado con `link` en un bloque de comentario condicional.

```
h3 { margin-bottom: 1.2em; }
```

```
li { margin: 0 0 -.214em 0; }
select { height: 1.429em; }
textarea { height: 4.571em; }
fieldset {
  height: 1.583em;
  padding-top: .417em;
}
.medium { width: 13.429em; }
select.medium { width: 13.714em; }
.long { width: 20.286em; }
fieldset .rInput { border: 0 !important; }
#subjectField { margin-bottom: -.214em; }
#availabilityField .rInput { margin-top: .286em; }
#messageField { padding-bottom: .286em; }
input.submitButton { margin-top: .15em; }
* html input, * html textarea { float: left; }
* html select { font-size: .643em; }
* html select.medium { width: 21.364em; }
* html textarea { height: 4.643em; }
* html #subjectField {
  margin-top: .071em;
  margin-bottom: 0;
}
* html #availabilityField label { padding-top: 0; }
* html input.submitButton { margin: .1em 0 0 7em; }
```

Archivo fuente de: "Página completa"
http://mosaic.uoc.edu/ac/le/operafiles/34/demo_form_complete.html

8.3.26. Demostración 12: consideraciones previas

- Como se puede observar, todos los estilos mostrados anteriormente sugieren ligeras diferencias en el modo como Internet Explorer transmite los tamaños de fuente por la cascada y dispone las cajas.

- Otro tema interesante es el selector `* html`. Internet Explorer 5 y 6 son los únicos navegadores que reconocen este selector, por lo que es un filtro de paso bajo efectivo para reglas de hoja de estilo dirigidas a estos navegadores.

8.4. Establecer niveles de soporte de plataforma

El último subapartado de demostraciones presenta el tipo de estilos reservados para Internet Explorer 6 y 7 y nos lleva a hablar de cómo un buen equipo de diseño trata navegadores de muy diferentes tipos.

La realidad de la web es que los usuarios utilizan una amplia variedad de navegadores en toda clase de entornos. Algunos son antiguos y otros son de rabiosa actualidad. Los hay que funcionan en ordenadores convencionales y los hay que funcionan en aparatos portátiles como teléfonos móviles. Todos ellos se desarrollan con sistemas operativos específicos y después se adaptan a otros con diferentes niveles de aceptación de estándares. Excepto Opera, todos los fabricantes de navegadores venden productos pensados para ser utilizados junto con otros títulos de una gran gama de productos, un requisito de producto que hace que los navegadores sean más complejos de lo necesario sólo para navegar por la Web.

Como si no hubiera bastante con pensar en las múltiples y variadas ventajas y desventajas de los navegadores, también se deben tener en cuenta los errores: errores de seguridad, errores de componentes y, sobre todo, los errores de representación. Los usuarios de Safari 3.x han descubierto que, en un punto, el documento de demostración descubre un molesto error de representación en su propio navegador.

La mejor manera de solucionar estos problemas es definir capas de soporte. El primero en promulgar este método fue el equipo de desarrollo de interfaz de Yahoo!, que lo denomina Graded Browser Support.

Por norma general, las capas de soporte se dividen en cuatro grandes categorías:

1) El sitio web se representa tal como se diseñó originalmente dentro de los límites de las capacidades de estos navegadores y acepta todas las funciones. La plataforma de desarrollo define lo que en ocasiones se llama el nivel "A+".

2) La desviación respecto de la composición preferente es evidente, quizá hasta un nivel destacado; sin embargo, el sitio web aún se puede utilizar y acepta la mayoría o todas las funciones del sitio.

3) El sitio web que se presenta a los usuarios de estos navegadores es el más sencillo que se puede hacer sin estropear la marca del propietario del sitio, y es posible que determinadas funciones del sitio queden inactivas. Estos navegadores tienen unas bases de instalación comparativamente pequeñas, y se los considera anticuados, inestables o faltos de funciones.

4) Este nivel de aceptación, que en la documentación de Yahoo! se denomina "X Grade" o de grado X, se reserva para las plataformas sin comprobar, es decir, habitualmente para los navegadores más nuevos con bases de instalación pequeñas (generalmente Opera, claro). Una vez probados, estos navegadores promocionan a una capa superior.

Los detalles del proceso de recopilación de requisitos que informan la definición de los niveles de soporte y asignan navegadores tienden a ser larguí-

simos y muy aburridos y, por lo tanto, se omiten de este apartado, que ya es bastante largo.

8.5. Composiciones de formulario complejas en la práctica (... en lugar de en teoría)

Entre las consideraciones previas proporcionadas anteriormente, se ha afirmado que las demostraciones se han organizado por el orden del código fuente de la hoja de estilo en lugar de por el orden en el que de hecho se añadieron las reglas a la hoja de estilo. Los motivos para hacerlo son:

- Para poder delimitar una serie de demostraciones cronológicamente, era necesario mantener un diario (o guardar la hoja de estilo en un sistema de control de versiones), cosa que no se hizo nunca. Cuando se detectó el despiste, ya hacía demasiado tiempo que había pasado la fecha límite de entrega del apartado para poder corregirlo.

- Secuenciar según el orden del código fuente hace que sea mucho más fácil producir los documentos de demostración; una concesión más del ideal a la práctica.

- Como la hoja de estilo de la demostración original se escribió yendo con un cuidado notable (si no total) para la normalización y el orden de las fuentes (como se debería hacer con todas las hojas de estilo de producción), secuenciar la demostración por orden de fuentes garantiza que los estudiantes que quieran "ver el código fuente" lo tendrán mucho más fácil para entender lo que se ofrece.

El programa Opera 9.6 para OS X fue el agente de usuario que se utilizó para el desarrollo; con esta advertencia y las anteriores, a continuación se presenta el orden general en que se hicieron los cambios y las añadiduras en la hoja de estilo:

1) Se aplican los estilos del documento (es decir, `body`).
2) Se restablecen los valores predeterminados de lista, formulario y `fieldset`.
3) Se especifica la tipografía.
4) Se limita y se aplica `clear` a los elementos de la lista.
5) Se aplican las `labels` en general.
6) Se especifica y normaliza la composición de los controles del formulario (sobre todo los tamaños).
7) Se crea el botón de enviar.
8) Se aplican los casos límite.
9) Se prueba Safari y Firefox y se cambian los valores de la hoja de estilo para reflejar los compromisos (allí donde se puede).

10) Se prueba Internet Explorer 6 y 7 y se les suministran ajustes de propiedad/valor en una hoja de estilo condicional.

El proceso que se acaba de describir empieza con las reglas más generales y va pasando a las más concretas hasta llegar a solucionar los pequeños problemas específicos de navegadores determinados... de manera muy parecida al orden del código fuente de la hoja de estilo en sí. Sin embargo, los resultados no se correlacionan a la perfección. Esto sucede porque los diferentes motores de representación y las peculiaridades de elementos como el contexto `float` conllevan consecuencias imprevistas cuando se mezcla todo, de modo que el proceso real va más allá de realizar unas cuantas comprobaciones, retoques y reconsideraciones.

Resumen

Este apartado proporciona una base completa para aplicar estilos y composición en los formularios, pero es posible ir más lejos todavía. Las dificultades que plantean los sistemas operativos (cuyos componentes se utilizan para crear los controles de formularios web) y las diferencias entre motores de representación aumentan el reto que supone para un especialista en estilos crear un formulario web según una serie de especificaciones. Este apartado deja la puerta abierta a la experimentación con la multitud de pequeños problemas asociados con esta tarea y muestra la manera de conseguir un buen nivel de dominio en uno de los aspectos más complicados del arte de diseñar webs.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿Cuál es el tipo de flujo de los controles de formulario que acepta datos introducidos por el usuario y cuáles son las dos características que los hacen destacables en cuanto a la composición?

2. ¿Qué dos atributos aparte de `value` (valor) y `disabled` (desactivado) manipulan la configuración de los controles de formulario de antemano de la interacción con el usuario y a qué elementos se aplican?

3. Este apartado de demostraciones os proporciona los campos obligatorios. Escribid como mínimo una regla de estilo que, de una vez, pueda cambiar el aspecto de un campo que incluya un error u omisión de introducción de datos del usuario.

4. Describid un caso de uso del elemento `select`, del control `checkbox` y del control `radio`. Confirmad cada descripción con una explicación de las ventajas que ofrece vuestra elección comparada con otras opciones posibles.

5. Utilizando una referencia en línea elegida por vuestro instructor, encontrad y describid brevemente alternativas a `input type="submit"`.

6. Cread un elemento `select` que permita seleccionar múltiples `options` añadiendo la pareja atributo/valor de `multiple="multiple"`. Después de examinar el comportamiento de este elemento, explicad por qué casi no se lo encuentra nunca en los lugares de producción.

Tabla: conversiones de fracciones a decimales

En la tabla siguiente, los dígitos que se incluyen entre paréntesis con un asterisco detrás se repiten hasta el infinito; por ejemplo, 0,2(6*) es equivalente a 0,266666666666666... (el 6 es periódico).

Los valores más próximos a cero se encuentran a la izquierda de la tabla y avanzan hacia el uno leyendo la tabla de izquierda a derecha.

x	1/x	2/x	3/x	4/x	5/x	6/x	7/x	8/x	9/x	10/x	11/x	12/x	13/x	14/x	15/x
2	0,5	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	0,(3*)	0,(6*)	-	-	-	-	-	-	-	-	-	-	-	-	-
4	0,25	0,5	0,75	-	-	-	-	-	-	-	-	-	-	-	-
5	0,2	0,4	0,6	0,8	-	-	-	-	-	-	-	-	-	-	-
6	0,1(6*)	0,(3*)	0,5	0,(6*)	0,8(3*)	-	-	-	-	-	-	-	-	-	-
7	0,(142857*)	0,(285714*)	0,(428571*)	0,(571428*)	0,(714285*)	0,(857142*)	-	-	-	-	-	-	-	-	-
8	0,125	0,25	0,375	0,5	0,625	0,75	0,875	-	-	-	-	-	-	-	-
9	0,(1*)	0,(2*)	0,(3*)	0,(4*)	0,(5*)	0,(6*)	0,(7*)	0,(8*)	-	-	-	-	-	-	-
10	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	-	-	-	-	-	-
11	0,(09*)	0,(18*)	0,(27*)	0,(36*)	0,(45*)	0,(54*)	0,(63*)	0,(72*)	0,(81*)	0,(90*)	-	-	-	-	-
12	0,08(3*)	0,1(6*)	0,25	0,(3*)	0,41(6*)	0,5	0,58(3*)	0,(6*)	0,75	0,8(3*)	0,91(6*)	-	-	-	-
13	0,(076923*)	0,(153846*)	0,(230769*)	0,(307692*)	0,(384615*)	0,(461538*)	0,(538461*)	0,(615383*)	0,(692307*)	0,(769230*)	0,(846153*)	0,(923076*)	-	-	-
14	0,0(714285*)	0,(142857*)	0,2(142857*)	0,(285714*)	0,3(571428*)	0,(428571*)	0,5	0,5(714285*)	0,6(428571*)	0,(714285*)	0,7(857142*)	0,(857142*)	0,9(285714*)	-	-
15	0,0(6*)	0,1(3*)	0,2	0,2(6*)	0,(3*)	0,4	0,4(6*)	0,5(3*)	0,6	0,(6*)	0,7(3*)	8	0,8(6*)	0,9(3*)	-
16	0,0625	0,125	0,1875	0,25	0,3125	0,375	0,4375	0,5	0,5625	0,625	0,6875	0,75	0,8125	0,875	0,9375

Lecturas complementarias

Bos; Bert y otros (2007). "Cascading style sheets level 2 revision 1 (CSS 2.1) specification". World Wide Web Consortium. [Fecha de consulta: 28 de mayo del 2008.]
http://www.w3.org/TR/2007/CR-CSS21-20070719

Henick, Ben (2006). "12 lessons for those afraid of CSS and standards". En: *A List Apart*. [Fecha de consulta: 16 de diciembre del 2008.]
http://www.alistapart.com/articles/12lessonsCSSandstandards

Horton, Sarah; Lynch, Patrick (2002). *Web style guide: basic principles for creating web sites* (2.ª ed.). New Haven, Conn.: Yale University Press. http://www.webstyleguide.com/

Meyer, Eric (2007). "Formal weirdness". *Meyerweb.com*. [Fecha de consulta: 17 de diciembre de 2008.] http://meyerweb.com/eric/thoughts/2007/05/15/formal-weirdness/

Raggett, Dave y otros (1999). "HTML 4.01 specification". World Wide Web Consortium. [Fecha de consulta: 30 de junio de 2008.] http://www.w3.org/TR/1999/REC-html401-19991224

Reynolds, Garr (2005). "From golden mean to 'rule of thirds'". Presentation Zen. [Fecha de consulta: 16 de diciembre de 2008.] http://www.presentationzen.com/presentationzen/2005/08/from_golden_-mea.html

Santa Maria, Jason (2008). "Making modular layout systems". *24 Ways*. [Fecha de consulta: 16 de diciembre de 2008.] http://24ways.org/2008/making-modular-layout-systems

Wikipedia. 2008. "Fahrner image replacement". [Fecha de consulta: 19 de diciembre de 2008.] http://en.wikipedia.org/wiki/Fahrner_Image_Replacement

9. Elementos flotantes y *clearing*

Tommy Olsson

En este apartado veremos qué son los elementos flotantes y el *clearing*, dos herramientas indispensables para el diseñador de webs modernas. Son unas herramientas muy versátiles que podéis utilizar para hacer que el texto fluya en torno a las imágenes o incluso crear composiciones de múltiples columnas.

9.1. ¿Para qué sirven `float` y `clear`?

Si miráis cualquier revista, veréis que hay imágenes que ilustran los apartados y que el texto las rodea y fluye a su alrededor. La propiedad `float` de CSS se creó para permitir este estilo de composición en las páginas web. Al hacer "flotar" una imagen, o cualquier otro elemento, ésta se desplaza hacia un lado y permite que el texto fluya por el otro. Aplicar *clearing* a un elemento flotante hace que éste se desplace hacia abajo, si es necesario, para evitar que aparezca justo al lado del texto.

Aunque en principio cualquier elemento puede ser un elemento flotante, los diseñadores utilizan esta propiedad básicamente para conseguir composiciones de múltiples columnas sin necesidad de abusar del etiquetado de tabla.

9.2. Algo de teoría muy aburrida

Para explicar cómo funcionan los elementos flotantes, primero deberemos dar un poco de teoría y ver cómo un navegador web muestra un documento HTML/CSS. No os preocupéis, será breve.

Todos los elementos HTML visibles generan una "caja" que inmediatamente se delimita. Si miráis el documento en una pantalla de ordenador o en un teléfono móvil, los cuadros aparecen en la pantalla. Si imprimís el documento, los cuadros aparecen en el papel. Si utilizáis un lector de pantalla, el contenido de los cuadros se reproduce oralmente.

De la misma manera que en HTML hay elementos de bloque y elementos en línea, en el CSS hay cajas de bloque y en línea. Por defecto, los elementos de bloque generan cajas de bloque y los elementos en línea generan cajas en línea. Aparte de las cajas generadas por los elementos, también se generarán otras cajas; por ejemplo, para el contenido textual del documento. Las cajas de bloque se presentan normalmente en el orden en el que aparecen los elementos en el etiquetado, de arriba abajo. Las cajas de bloque no pueden aparecer unas al lado de otras si no aplicamos CSS. Las cajas en línea se distribuyen ho-

rizontalmente. La propiedad `direction` determina si se distribuyen de izquierda a derecha o de derecha a izquierda (si no se especifica esta propiedad, estas cajas se distribuyen por defecto de izquierda a derecha).

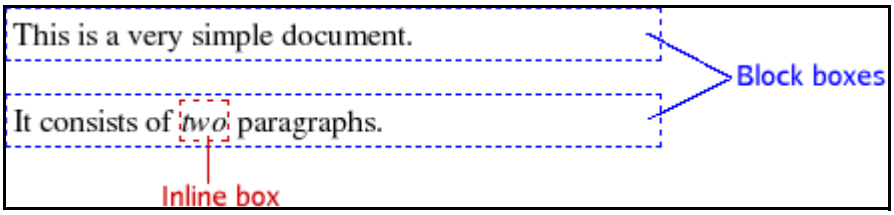
Esto se conoce como el flujo de documento: las cajas en línea fluyen horizontalmente dentro de sus cajas de bloque madre, y las cajas de bloque fluyen verticalmente. Las cajas aparecen en el mismo orden que los elementos del etiquetado HTML.

Pongamos por caso el siguiente documento HTML, que es muy básico (sólo hemos incluido la parte que hay dentro del elemento `body`):

```
<p>This is a very simple document.</p>
<p>It consists of <em>two</em> paragraphs.</p>
```

La figura 1 muestra una captura de pantalla de este documento con una sobreimpresión que muestra las dos cajas de bloque generadas por los elementos `p` y la caja en línea generada por el elemento `em`.

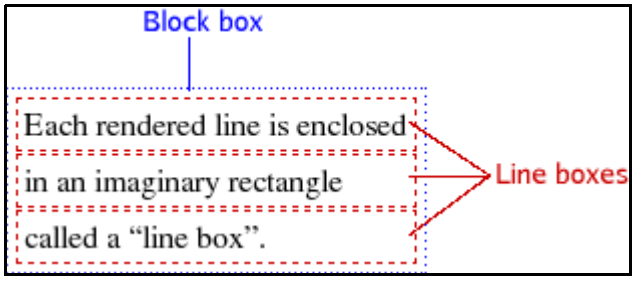
Figura 1



Muestra de las cajas de bloque generadas por los elementos `p` y de la caja en línea generada por el elemento `em`

Todas las cajas en línea que forman una "línea" en el dispositivo de salida están rodeadas por rectángulos imaginarios que se conocen como cajas de línea. Las cajas de línea se distribuyen siempre de arriba abajo sin ningún espacio entre ellas, tal como podéis ver en la figura 2.

Figura 2



Cada línea representada está cerrada en una caja de línea independiente.

9.3. ¿Cómo funcionan los elementos flotantes?

Ahora que ya hemos visto todas estas cuestiones teóricas tan aburridas, pasemos a ver la sintaxis de los elementos `float` y `clear` y a ver algunos ejemplos.

La propiedad `float` tiene cuatro valores válidos: `left`, `right`, `none` e `inherit`. Los dos primeros valores son de lejos los que se utilizan más habitualmente y hacen que una caja flote hacia la derecha o hacia la izquierda. La declaración `float:none`, que es el valor por defecto, se utiliza normalmente sólo para "deshacer" una declaración de alguna otra regla. El uso de `float:inherit` no es nada habitual; no lo hemos visto nunca en ningún sitio y probablemente sólo existe por una cuestión de coherencia. Hace que el elemento herede el valor de `float` de su elemento padre.

Una caja flotante se extrae del flujo del documento y se desplaza todo lo posible hacia la izquierda o hacia la derecha, según la dirección de flotación especificada. "Todo lo posible" significa normalmente hasta que el borde exterior del elemento flotante toca el borde del bloque que lo contiene (el interior de su relleno, si está definido). Así pues, en el caso de `float:left`, la caja se desplaza hacia la izquierda hasta que el margen izquierdo del elemento flotante toca el borde izquierdo del padre.

Los lectores que hayan estado alerta pueden haber visto que hemos ido utilizando "normalmente". Si ya hay una caja flotada hacia la izquierda cuando flotamos otra en esta misma dirección, la segunda caja se detendrá cuando toque a esta primera. Es decir, que los elementos flotantes no se pueden poner los unos sobre los otros.

Ya ha llegado el momento de ver los elementos flotantes en acción, de manera que ya podéis preparar vuestro editor de textos.

1. Cread un archivo nuevo, copiad el código siguiente y guardad el documento como `float.html`.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset=utf-8">
      <title>Floating</title>
   </head>
    <body>
      <p id="p1"><span id="span-a">Lorem ipsum</span>
      <span id="span-b">dolor sit amet</span>
      <span id="span-c">consectetuer</span> adipiscing elit.
      Curabitur feugiat feugiat purus.
      Aenean eu metus. Nulla facilisi.
      Pellentesque quis justo vel massa suscipit sagittis.
      Class aptent taciti sociosqu ad litora torquent per conubia nostra, per
      inceptos hymenaeos.
```

```
      Quisque mollis, justo vel rhoncus aliquam, urna tortor varius lacus, ut
      tincidunt metus arcu vel lorem.
       Praesent metus orci, adipiscing eget, fermentum ut, pellentesque non, dui.
       Sed sagittis, metus a semper dictum, sem libero sagittis nunc, vitae
      adipiscing leo neque vitae tellus.
       Duis quis orci quis nisl nonummy dapibus.
       Etiam ante. Phasellus imperdiet arcu at odio.
       In hac habitasse platea dictumst. Aenean metus.
       Quisque a nibh. Morbi mattis ullamcorper ipsum.
       Nullam odio urna, feugiat sed, bibendum sed, vulputate in, magna.
      Nulla tortor justo, convallis iaculis, porta condimentum, interdum nec, arcu.
       Proin lectus purus, vehicula et, cursus ut, nonummy et, diam.</p>
       <p id="p2">Nunc ac elit. Vestibulum placerat dictum nibh. Proin massa.
       Curabitur at lectus egestas quam interdum mollis.
       Cras id velit a lacus sollicitudin faucibus.
       Proin at ante id nisi porttitor scelerisque.
       In metus. Aenean nonummy semper enim.
       Aenean tristique neque quis arcu tincidunt auctor.
       Fusce consequat auctor ligula.
       Fusce nulla lorem, sagittis a, lacinia et, nonummy in, eros.
       In nisi augue, aliquam eget, convallis vel, malesuada quis, libero.</p>
       <p id="p3">Hello, World!</p>
    </body>
</html>
```

Ya sé que es mucho contenido, pero lo necesitamos para ver bien cómo funciona.

2. Abrid el documento en vuestro navegador web para ver qué aspecto tiene. Aburrido, ¿verdad?

3. Cread otro documento con el editor de textos, copiad el código siguiente y guardadlo con el nombre `style.css` en el mismo directorio que el fichero HTML del paso 1.

```
#span-a {
  float: left;
  background-color: #cfc;
  color: #030;
}
```

4. Enlazad la hoja de estilo al documento HTML insertando la línea siguiente justo antes de la etiqueta `</head>`.

```
<link rel="stylesheet" type="text/css" href="style.css">
```

5. Guardadlo y actualizad la página en el navegador. Veréis que el elemento `span` que contiene las palabras "Lorem ipsum" se ha desplazado flotando hacia la izquierda. También hemos puesto un fondo de color verde claro para hacer que destaque un poco.

6. Todavía no es fácil ver qué sucede aquí, o sea que haremos que el elemento flotante sea un poco mayor. Añadid la siguiente declaración a vuestra hoja de estilos:

```
#span-a {
  float: left;
  background-color: #cfc;
  color: #030;
  padding: 1em;
}
```

7. Guardad y actualizad, y veréis que el área de color ahora es un poco mayor, ya que hemos añadido un poco de relleno por los cuatro lados del cuadro. El elemento flotante ocupa la altura de tres líneas de texto y podemos ver claramente que el otro texto fluye en torno a este elemento flotante.

9.3.1. Los detalles

Ahora analizaremos con más detalle qué es lo que sucede realmente aquí. La caja flotante generada por el primer elemento `span` se ha desplazado hacia la izquierda, hasta el borde del documento, y las cajas de línea adyacentes se han reducido. Aunque todavía no es visible, la caja de bloque generada por el párrafo que contiene el elemento flotante se ve afectada. Ahora destacaremos el párrafo para que quede más claro.

1. Añadid la siguiente regla CSS a la hoja de estilos:

```
p {
  border: 1px solid #f00;
}
```

2. Volved a guardar el archivo CSS y actualizad la ventana del navegador. Ahora deberíais ver un borde rojo en torno a cada uno de los párrafos; observad que el elemento flotante se encuentra dentro de uno de los párrafos.

3. Ahora modificaremos la última regla para verificar que el elemento flotante se detiene cuando llega al borde interior del área de separación del padre:

```
p {
  border: 1px solid #f00;
  padding: 1em;
  background-color: #ff9;
}
```

4. Guardad y actualizad, y veréis la prueba de lo que os hemos dicho antes: el cuadro flotante se desplaza hasta cerca del bloque que lo contiene, mientras que la separación del padre queda fuera de éste. También veréis que el fondo amarillo del párrafo se extiende por debajo de la caja flotante. Claramente, el hecho de flotar una caja hija no afecta en absoluto a la caja del párrafo, sólo a las cajas de línea que hay en su interior.

5. Ahora haremos unos cuantos experimentos más: ¿qué ocurre si el elemento flotante es más alto que su padre? Modificad la regla para el elemento flotante de la manera siguiente:

```
#span-a {
  float: left;
  background-color: #cfc;
  color: #030;
  padding: 1em 1em 10em;
}
```

Nota

Si la ventana del navegador es estrecha, es posible que debáis utilizar un valor mayor que 10 em para la separación inferior con el fin de que el área verde se extienda más allá del borde inferior del párrafo.

Ahora veréis algo muy interesante: la caja flotante sobresale fuera de la caja madre; la caja madre no se amplía para poder contener a su caja hija flotante. También podéis ver (si habéis utilizado una separación inferior lo bastante grande) que las cajas de línea adyacentes al elemento flotante del segundo párrafo se han acortado.

9.3.2. Más elementos flotantes

Ahora crearemos otro elemento flotante para ver qué sucede cuando se desplazan dos elementos flotantes en la misma dirección.

1. Añadid una regla nueva a la hoja de estilos, guardad y actualizad igual que habéis hecho antes.

```
#span-b {
  float: left;
  background-color: #ccf;
  color: #003;
  padding: 1em;
}
```

El elemento `span` que contiene las palabras "dolor sit amet" también se ha desplazado flotando hacia la izquierda. Veréis que se ha desplazado hacia la izquierda hasta tocar el primer elemento flotante; es decir, "todo lo posible".

2. ¿Y por qué sólo dos elementos flotantes? Crearemos aún un tercero. Añadid la regla siguiente a vuestra hoja de estilos:

```
#span-c {
  float: left;
  background-color: #fcc;
  color: #300;
  padding:2em 1em;
}
```

3. También queremos añadir una regla temporal para ver un ejemplo de qué ocurre cuando no hay suficiente espacio para un elemento flotante en una línea. Añadid la regla siguiente al final de la hoja de estilos:

```
span {
  width: 34%;
}
```

4. Igual que antes, guardad vuestra hoja de estilos y actualizad el documento en el navegador; veréis algo similar a lo que hay en la figura 3.

Figura 3

¿No es exactamente lo que esperabais?

¡Mira por dónde! ¿Qué ha pasado? El tercer elemento flotante aparece ahora bajo el segundo. (Y, además, Internet Explorer 6 hace otras cosas extrañas, que de momento ignoraremos). Como la anchura de cada uno de los elementos `span` es el 34% de la anchura del párrafo (como especifica la regla añadida en el paso 3), más un poco de separación, no hay suficiente espacio para los tres uno al lado del otro (3 × 34% = 102%). Los dos primeros elementos flotantes caben en la misma línea, pero el tercero no y se desplaza hacia abajo. Lo que es importante es que se desplaza hacia abajo sólo lo necesario para caber en la línea. No se desplaza bajo el primer elemento flotante más alto porque hay espacio suficiente a su derecha.

Otra cosa interesante que hay que tener en cuenta es que habéis asignado una anchura a los elementos `span`. Esto no debería representar ninguna diferencia porque `span` es un tipo de elemento en línea. No obstante, cuando se hace flotar una caja, ésta se convierte automáticamente en caja de bloque, lo que significa que le podemos asignar dimensiones y márgenes verticales.

9.3.3. Márgenes en los elementos flotantes

Ahora veremos lo que se puede hacer con los márgenes de los elementos flotantes.

1. En primer lugar, eliminad la regla temporal para los elementos `span` que habéis añadido antes y después guardad y actualizad, de manera que los tres elementos flotantes vuelvan a quedar uno al lado del otro. Es decir, borrad esta regla:

```
span {
  width: 34%;
}
```

Ahora los tres elementos flotantes están totalmente juntos y el texto adyacente empieza inmediatamente después del último elemento flotante (a no ser que utilicéis Microsoft Internet Explorer 6 o una versión anterior, ya que en este caso habrá un espacio de 3 píxeles a la derecha a causa del problema de los tres píxeles*). ¿Cómo se puede crear un poco de espacio en torno a un cuadro flotante? La respuesta son los **márgenes**.

* http://positioniseverything.net/explorer/threepxtest.html

2. Lo probaremos con el elemento flotante central; cambiaremos la regla CSS para el elemento flotante central de la manera siguiente y entonces guardaremos y actualizaremos:

```
#span-b {
  float: left;
  background-color: #ccf;
  color: #003;
  padding: 1em;
  margin-left: 1em;
  margin-right: 1em;
}
```

Ahora sí que hay un poco de espacio a ambos lados del elemento flotante central.

3. También podéis definir márgenes verticales para una caja flotante; haced los cambios siguientes en la regla para el tercer elemento flotante, guardad y actualizad.

```
#span-c {
  float: left;
  background-color: #fcc;
  color: #300;
  padding:2em 1em;
  margin-top: 2em;
  margin-bottom: 2em;
}
```

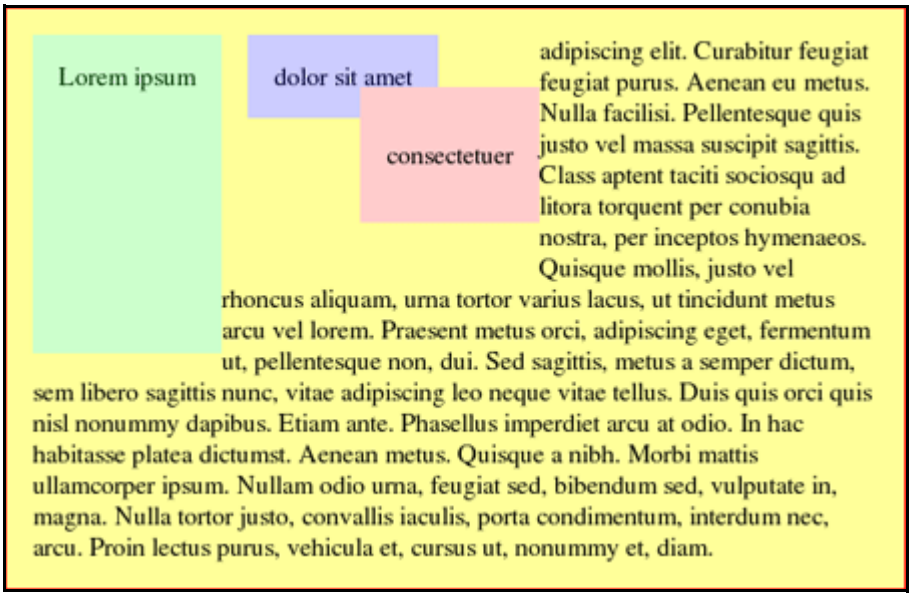
El tercer elemento flotante se ha desplazado hacia abajo y también hay un poco de espacio adicional debajo.

4. Como ya vamos lanzados hacia la aventura, veamos qué ocurre si empezamos a jugar con márgenes negativos. Haced los cambios siguientes en la regla para el tercer elemento flotante, guardad y actualizad:

```
#span-c {
  float: left;
  background-color: #fcc;
  color: #300;
  padding:2em 1em;
  margin-top: 2em;
  margin-bottom: 2em;
  margin-left: -4em;
}
```

Ahora veréis el resultado que se muestra en la figura 4:

Figura 4



¡Ahora los elementos flotantes aparecen unos sobre otros!

¿Cómo puede ser? ¿Quién ha dicho que los elementos flotantes no pueden aparecer los unos sobre los otros? El margen negativo desplaza todo el elemento flotante hacia la izquierda.

El uso de márgenes negativos en los elementos flotantes puede ser muy útil para crear algunos tipos de composiciones de múltiples columnas.

9.4. *Clearing*

Ahora que ya hemos visto las bases de los elementos flotantes, podemos pasar a otro tema muy relacionado: el *clearing*.

Como hemos visto en los ejemplos a lo largo de este apartado, el texto fluye en torno a un elemento flotante y las cajas de bloque no se ven afectadas por los elementos flotantes. Algunas veces es preferible que un elemento no acabe colocado justo al lado de un elemento flotante. Por ejemplo, un título que introduce una sección nueva de un apartado no debe aparecer al lado de una imagen de la sección previa. Es mucho mejor hacer que el título aparezca bajo la imagen, incluso aunque la imagen sobresalga por debajo del último párrafo. La única manera de hacerlo es usar la propiedad `clear` (distanciar) en el título.

Otro ejemplo de esto es la omnipresente composición de tres columnas con un pie de página que ocupa toda la anchura. Si las columnas son flotantes, entonces utilizaréis la propiedad `clear` en el pie para garantizar que aparezca bajo todas las columnas, sea cual sea la más larga.

La propiedad `clear` tiene tres valores útiles: `left` (izquierda), `right` (derecha) y `both` (ambos). También son válidos los valores `none` (ninguno), que es el valor por defecto, e `inherit` (heredar).

El uso de `clear:left` en un elemento garantiza que su caja generada aparezca bajo cualquier caja flotada previamente hacia la izquierda. Si utilizáis `clear:both`, aparecerá bajo todos los elementos flotantes previos de ambos lados.

La distancia se consigue desplazando el elemento hacia abajo (el espacio en blanco se añade sobre su margen superior), si es necesario, hasta que su borde superior se encuentre bajo los bordes inferiores de todos los cuadros flotantes en la o las direcciones especificadas. Veamos un ejemplo para ilustrarlo mejor.

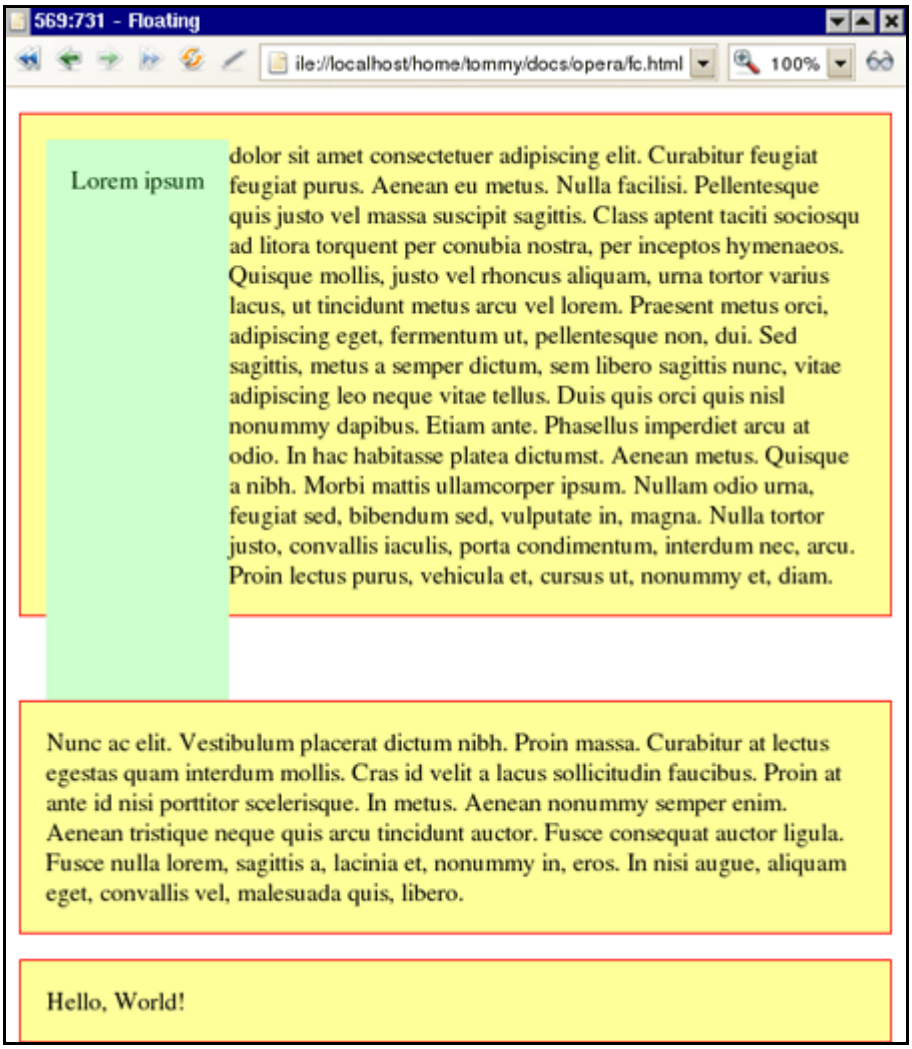
1. Antes de hacer la prueba, limpiemos primero la hoja de estilos. Eliminad las reglas para `#span-b` y `#span-c` de manera que quede sólo el elemento flotante verde de la izquierda. Comprobad que su separación inferior sea lo suficientemente grande como para extenderse dentro del segundo párrafo.

2. Añadid la regla siguiente para el segundo párrafo, guardad y actualizad:

```
#p2 {
  clear: left;
}
```

¡Mirad! El segundo párrafo se desplaza hacia abajo hasta que queda por debajo del elemento flotante, tal como podéis ver en la figura 5.

Figura 5



Ahora el segundo párrafo queda por debajo del primero.

Para complicar todavía más las cosas, podemos utilizar `float` y `clear` en el mismo elemento.

3. Añadid una regla al segundo elemento flotante para separarlo del primero, guardad y actualizad:

```
#span-b {
  float: left;
  clear: left;
  padding: 1em;
  background-color: #ccf;
  color: #003;
}
```

El elemento flotante azul aparece ahora bajo el elemento flotante verde, totalmente fuera del párrafo padre. Como también se ha flotado hacia la izquierda, el segundo párrafo se desplaza un poco más hacia abajo para quedar separado.

9.5. Contener elementos flotantes

Como ya hemos visto antes, la caja madre no se amplía normalmente para contener a los hijos flotantes. Esto a menudo puede provocar confusiones, por ejemplo cuando todos los hijos de un elemento flotan al crear un menú horizontal a partir de una lista no ordenada mediante la flotación de todos los elementos `li`. Como las cajas flotantes se sacan del flujo y no afectan a la caja madre, el hecho de hacer flotar las hijas hace que en realidad la madre quede vacía y que pase a tener una altura cero. Algunas veces esto no es lo que queremos, por ejemplo a la hora de definir un fondo para la madre. Si la madre tiene una altura cero, el fondo no se verá.

Es evidente que necesitamos algún mecanismo para conseguir que una caja madre se amplíe para incluir a sus hijas flotantes. El método tradicional consistía en incluir un elemento adicional en el etiquetado, justo antes de la etiqueta de cierre de la caja madre, y definir en él `clear:both`. Esto funciona, pero no es muy aceptable porque implica introducir un etiquetado adicional poco semántico e innecesario. Por suerte, hay otras maneras que explicaremos a continuación.

El primer método consiste sencillamente en hacer flotar también a la madre. Las cajas flotantes se amplían siempre para incluir todos sus cajas hijas.

1. Para probarlo en nuestro documento de ejemplo, eliminad otra vez la regla para `#span-b`, haced flotar el primer párrafo de la manera siguiente y guardad y actualizad:

```
#p1 {
  float: left;
}
```

El párrafo se amplía ahora hasta incluir el elemento flotante de color verde. Todo eso está muy bien, pero algunas veces no tenemos la opción de flotar a la madre. Otra manera de hacer lo mismo sin flotar a la caja madre es definiendo la propiedad `overflow` (desbordamiento) de la madre a un valor diferente de `visible`. Si definís el valor `hidden` y no especificáis ninguna altura, la caja madre incluirá las cajas hijas flotantes.

2. Sustituid la última regla por la siguiente, guardad y actualizad:

```
#p1 {
  overflow: hidden;
}
```

Tened en cuenta que el último método no funciona con Internet Explorer 6 o anterior.

9.6. Ajustar

Ya hemos comentado antes que el hecho de hacer flotar una caja en línea hacía que se convirtiera en una caja de bloque, lo que permite especificar sus dimensiones y márgenes verticales. Hacer flotar una caja de bloque también tiene una consecuencia sorprendente: si no se especifica la anchura de la caja, ésta "se ajusta" para adaptarse a su contenido. Esto no era visible en el documento de ejemplo cuando habéis hecho flotar el primer párrafo porque ya tenía suficiente contenido para llenar toda la ventana (a no ser que tuvierais un monitor realmente ancho).

Ahora haremos flotar el último párrafo para ver el efecto. De hecho, sólo para tener un poco de variación, haremos una locura y lo haremos flotar a la derecha.

Añadid la regla siguiente en la hoja de estilos, guardad y actualizad:

```
#p3 {
  float: right;
}
```

El párrafo que dice "Hello, World!" flotará hacia la derecha y tendrá sólo la anchura del texto, más un poco de relleno que habéis especificado en una regla anterior para todos los párrafos.

9.7. Centrar elementos flotantes

Algunas veces querréis hacer flotar un elemento, quizá para que incluya hijos flotantes, pero a la vez mantenerlo centrado horizontalmente dentro de su padre. Esto representa un problema: no es posible utilizar el truco habitual de definir los márgenes izquierdo y derecho en `auto` para los elementos flotantes, y no existe ningún valor `float:center`. ¿Hay alguna manera de solucionarlo?

Sí, la hay. El gurú del CSS Paul O'Brien explica cómo hacerlo en su artículo *When is a float not a float?**. Implica el uso de un elemento de envoltorio adicional, pero es perfectamente asumible. El principio utiliza el posicionamiento relativo, que explicaremos en el apartado siguiente. Desplazando el elemento de envoltorio hacia la derecha y a continuación el elemento flotante hacia la izquierda, se puede llegar a centrar un elemento flotante ajustado de anchura desconocida. (Podéis utilizar este conocimiento para impresionar a vuestra pareja en vuestra próxima cita. No falla nunca.)

*** http://www.search-this.com/2007/09/19/when-is-a-float-not-a-float/**

Podéis ver el apartado 10 de este módulo.

Intentémoslo. En el ejemplo siguiente añadiremos una barra de menú horizontal a vuestra página. El menú estará basado en una lista no ordenada con elementos flotantes.

1. Insertad el etiquetado siguiente justo después de la etiqueta `<body>` de vuestro documento HTML:

```
<div class="wrap">
  <ul id="menu">
    <li><a href="#">Home</a></li>
    <li><a href="#">News</a></li>
    <li><a href="#">Products</a></li>
    <li><a href="#">Services</a></li>
  </ul>
</div>
<!—Necesitamos esto para Internet Explorer-->
<div class="clear"></div>
```

2. Añadid las reglas siguientes de CSS en la hoja de estilos para aplicar estilos al menú:

```
#menu {
  margin: 0;
  padding: 0.5em;
  font-family: Verdana,sans-serif;
}
#menu li {
  float: left;
  list-style-type: none;
  margin: 0 0 0 0.5em;
  padding: 0.25em;
  background-color: #600;
  color: #ff9;
  border: 2px solid #f00;
}
#menu a {
  color: #ff9;
  text-decoration: none;
}
.wrap {
  float: left;
  margin-bottom: 2em;
}
```

```
.clear {
  clear: left;
  height: 1px;
  margin-top: -1px;
}
```

3. Guardad los dos archivos y actualizad el navegador. Veréis el menú en la parte superior izquierda. Ahora lo centraremos horizontalmente.

4. Desplazad el elemento de envoltorio hasta el punto central modificando la regla de `.wrap` como se indica a continuación:

```
.wrap {
  float: left;
  margin-bottom: 2em;
  position: relative;
  left: 50%;
}
```

El menú empezará en el centro horizontal de la página, pero esto no es lo que queríamos; está demasiado desplazado hacia la derecha, o sea que lo deberemos desplazar un poco hacia la izquierda. Como habéis flotado el elemento de envoltorio, éste se ha ajustado para adaptarse a la lista. Debéis desplazar la lista una distancia equivalente a la mitad de su anchura, lo que también significa la mitad de la anchura del envoltorio, o sea que lo deberemos desplazar un -50%.

5. Modificad la regla `#menu` de la manera siguiente:

```
#menu {
  margin: 0;
  padding: 0.5em;
  font-family: Verdana,sans-serif;
  position: relative;
  left: -50%;
}
```

El menú ya está centrado; el único problema es que puede haber una barra de desplazamiento horizontal según la anchura de la lista y de la ventana del navegador. Esto ocurre porque habéis desplazado el elemento de envoltorio hasta el centro de la pantalla; si la lista es más ancha que la mitad de la ventana, una parte de ésta quedará fuera.

6. Podéis evitarlo definiendo `overflow:hidden` en un elemento padre adecuado para ocultar el desbordamiento. En este caso, el padre del envoltorio es el `body`. Algunas veces no es factible ocultar el desbordamiento del elemento `body`, y en este caso necesitaréis un envoltorio para el envoltorio; aquí, sin embargo, todo funciona correctamente.

Añadid la regla siguiente a vuestra hoja de estilos:

```
body{
  overflow: hidden;
}
```

7. De hecho, aún hay otro problema. Si lo abrís con Internet Explorer, veréis que aún no funciona del todo bien. La manera de solucionarlo es hacer flotar la lista misma, pero sólo en Internet Explorer, ya que en otros navegadores quedaría rota. Podéis solucionarlo utilizando un pequeño truco que garantiza que esta regla sólo se aplicará en Internet Explorer.

Añadid la regla siguiente a vuestra hoja de estilos:

```
* html #menu {
  float: left;
}
```

9.8. ¡Errores!

Los elementos flotantes y el *clearing* son muy útiles, pero por desgracia la mayoría de los navegadores (de hecho, casi todos) aplica estas propiedades de una manera errónea. Internet Explorer 6 presenta una serie increíble de comportamientos extraños con los elementos flotantes, que incluyen la desaparición del contenido, márgenes dobles y el famoso problema de los tres píxeles. Pero cuando se trata de aplicar los elementos flotantes y el *clearing*, ni siquiera Firefox ni Opera están totalmente libres de problemas. *Position Is Everything** es un recurso muy valioso en el que se documentan todos estos problemas, junto con soluciones para la mayoría de los casos.

*** http://positioniseverything.net/**

Resumen

El hecho de hacer flotar una caja provoca que ésta se desplace todo lo posible hacia la izquierda o la derecha dentro de su elemento padre. Una caja flotante se extrae del flujo del documento y no afecta a la caja madre ni a las cajas de

bloque siguientes, aunque las cajas de línea adyacentes se acortan. Cuando no hay espacio para una caja flotante en una línea a causa de otros elementos flotantes previos, ésta se desplaza hacia abajo hasta que quepa (o hasta que no haya otros elementos flotantes).

Cuando se hace flotar una caja en línea, ésta se convierte en una caja de bloque. Al hacer flotar una caja de bloque y no se especifica ninguna anchura concreta, ésta se ajusta para adaptarse a su contenido.

Para centrar elementos flotantes hay que desplazar el contenido hacia abajo, si es necesario, hasta que su borde superior se encuentre por debajo de los bordes inferiores de todas las cajas flotantes en la dirección especificada.

Para centrar una caja flotante contenida se puede añadir un elemento de envoltorio y utilizar juiciosamente el posicionamiento relativo.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿Qué sucede si flotáis un elemento en medio de un párrafo, es decir, si hay texto antes del elemento flotante? Probadlo en navegadores diferentes, porque se comportan de maneras diferentes. Opera y Safari lo hacen bien, pero Firefox e Internet Explorer no.

2. ¿Cómo se pueden utilizar los elementos flotantes para mostrar miniaturas de imágenes en una galería como "celdas" del mismo tamaño sin utilizar una tabla para la distribución?

3. ¿Cómo podéis tener un menú de navegación vertical a la izquierda de la página y una columna de contenido a la derecha sin que el texto del contenido rodee el menú por debajo?

4. Una composición web muy habitual consiste en una cabecera que ocupa toda la anchura, tres columnas de contenido debajo de ésta y a continuación un pie de página al final que vuelve a ocupar toda la anchura. ¿Cómo se puede conseguir esta composición con elementos flotantes y *clearing*?

10. Posicionamiento estático y relativo con CSS

Tommy Olsson

En este apartado explicaremos con exhaustividad el modo como se puede utilizar CSS para posicionar elementos HTML en el lugar deseado de una página con la propiedad `position` (posicionar) de CSS y algunas propiedades relacionadas.

La propiedad `position` de CSS tiene cuatro valores lícitos (aparte del omnipresente `inherit`): `static` (estático), `relative` (relativo), `absolute` (absoluto) y `fixed` (fijo). Estos valores tienen un impacto muy importante sobre la manera en que se representa un elemento. Los valores `static` y `relative` están muy relacionados, y en este apartado los estudiaremos con gran detalle. Los valores `absolute` y `fixed` también están muy relacionados, pero de momento los dejaremos para el próximo apartado.

10.1. El maravilloso mundo de los rectángulos

Podéis ver el apartado 9 de este módulo.

Para empezar, recapitularemos un poco todo lo que hemos dicho en el apartado anterior, dedicado a los elementos flotantes y al *clearing*, sobre el CSS y las cajas HTML. Un documento HTML consiste en un número de elementos salpicados con datos de caracteres (texto). Cuando un documento de este tipo se reproduce en una pantalla de ordenador o en una copia impresa, estos elementos generan cajas rectangulares. De la misma manera que un conjunto de elementos HTML se divide en elementos de bloque y elementos en línea, las cajas CSS también pueden ser básicamente cajas de bloque o cajas en línea. Por defecto, la hoja de estilos del agente de usuario integrado de un navegador hace que los elementos HTML de bloque, como `p` y `div`, generen cajas de bloque, mientras que los elementos en línea, como `strong` y `span`, generan cajas en línea. Podemos utilizar la propiedad `display` para controlar el tipo de caja que se generará.

*** http://www.w3.org/TR/CSS21**

Las cajas generadas por los elementos de un documento se disponen según una serie de normas claramente definidas según la especificación CSS2.1*****. Estas reglas están escritas para las relativamente escasas personas que escriben software para navegadores con el fin de que puedan saber cómo funciona el CSS, pero no para aquéllos de nosotros que diseñamos páginas web para ganarnos la vida o como afición. ¡Por ello existe todo este curso! Como resultado, la especificación puede ser un tanto difícil de entender. En este apartado, intentaremos explicar los conceptos básicos de una manera que resulte apropiada para los diseñadores y desarrolladores de webs.

10.2. Posicionamiento estático

De hecho, este nombre no es nada adecuado. En realidad, los cuadros con `position:static` no se "posicionan" en el sentido del CSS. Simplemente se

disponen en el orden en el que aparecen en el etiquetado y ocupan todo el espacio que necesitan; éste es el comportamiento por defecto que se da cuando no se aplica ningún CSS al HTML.

Existen algunas diferencias fundamentales en la manera como se distribuyen las cajas de bloque en comparación con la distribución de las cajas en línea; por tanto, examinaremos los dos tipos uno a uno. Empezaremos con las cajas de bloque porque son más sencillas.

10.2.1. Disposición de cajas de bloque

Si no aplicamos ninguna declaración CSS concreta, las cajas de bloque se disponen verticalmente de arriba abajo en el orden en el que aparecen en el etiquetado. Cada caja es normalmente tan ancha como el documento (el elemento `body`), pero incluso si las hacemos más estrechas, éstas no se distribuirán una al lado de la otra aunque haya espacio, sino que se seguirán situando la una bajo la otra. Podéis imaginároslo como si cada caja de bloque tuviera un salto de línea implícito antes y después, ya que así se garantiza que tendrá una "línea" propia.

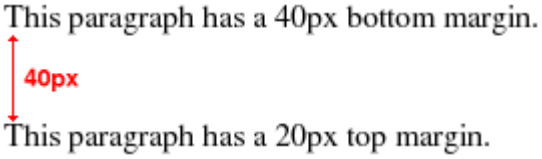
La distancia vertical entre dos cajas de bloque se controla con la propiedad `margin-bottom` (margen-inferior) de la primera caja y la propiedad `margin-top` (margen-superior) de la segunda caja (en esta asignatura ya hemos visto cómo manipular estas propiedades). Para las cajas del flujo normal, es decir, las cajas que no son flotantes o que no tienen un posicionamiento absoluto, los márgenes verticales entre dos cajas de bloque adyacentes se superpondrán, de manera que el resultado final no será la suma de los dos márgenes, sino el mayor de los dos, tal como se puede ver en la figura 1 que se muestra más adelante.

Mirad el siguiente fragmento de HTML:

```
<p style="margin-bottom:40px">This paragraph has a 40px bottom margin.</p>
<p style="margin-top:20px">This paragraph has a 20px top margin.</p>
```

Cuando se muestra en un navegador, los márgenes se superponen como se puede ver en la figura 1.

Figura 1

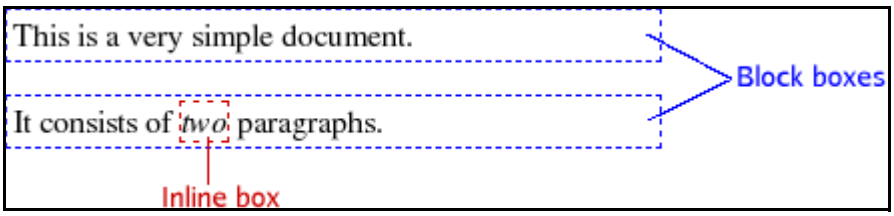


Los márgenes se superponen y la distancia entre los dos es de 40 píxeles, y no de 60 píxeles.

Una caja de bloque incluirá sólo otras cajas de bloque o sólo cajas en línea. Si un elemento de bloque contiene una mezcla de hijos de bloque y en línea (algo permitido pero semánticamente cuestionable), se generará lo que se conoce como *cajas de bloque anónimas* para incluir las cajas hijas insertadas, de manera que la madre contenga sólo cajas de bloque.

Podéis especificar las dimensiones de una caja de bloque con las propiedades `width` (anchura) y `height` (altura). También podéis especificar los márgenes vertical y horizontal. El valor inicial (por defecto) para `width` y `height` es `auto`, y el valor inicial para las propiedades de margen es `0`. Estos factores combinados significan que una caja de bloque será por defecto tan ancha como su madre, como muestra la figura 2.

Figura 2



Las cajas de bloque se distribuyen verticalmente.

10.2.2. Disposición de cajas en línea

> **Nota**
>
> Este subapartado puede ser difícil de entender si no tenéis mucha experiencia con CSS, de modo que quizá deberéis leerlo unas cuantas veces, pero tampoco os debe preocupar. La experimentación por vuestra cuenta es probablemente la mejor manera para entender bien todas estas cuestiones; sólo es necesario que a la hora de hacer las pruebas utilicéis un buen navegador compatible con los estándares, como Opera o Firefox.

Las cajas en línea se generan por defecto a partir de los elementos HTML en línea, pero también hay cajas en línea anónimas generadas para incluir el contenido de texto de los elementos.

Las cajas en línea se distribuyen horizontalmente, una después de otra, en el orden en el que aparecen en el etiquetado. Según la propiedad `direction`, los cuadros insertados se distribuirán de izquierda a derecha (`direction:ltr`) o de derecha a izquierda (`direction:rtl`). La dirección de izquierda a derecha se utiliza, por ejemplo, con los idiomas europeos, mientras que la de derecha a izquierda se utiliza con idiomas como el árabe y el hebreo.

El grupo de cajas en línea que forman una línea en la pantalla (o en el papel) está contenido en otro rectángulo más, que se conoce como *caja de línea*. Las cajas de línea se distribuyen verticalmente en su bloque madre, sin ningún es-

pacio entre ellas. Podemos modificar la altura de las cajas de línea con la propiedad `line-height`.

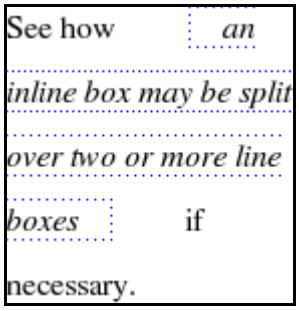
Para las cajas en línea no podemos especificar ninguna dimensión. Podemos especificar los márgenes horizontales, pero no los verticales.

Si es necesario, una caja en línea se puede dividir en varias cajas en línea distribuidas por dos o más cajas de línea. Cuando se produce una división de este tipo, todos los márgenes horizontales y separaciones, y todos los bordes verticales, se aplicarán sólo antes de la primera caja y después de la última caja. Imaginémonos un documento con la regla siguiente para los elementos `em`:

```
em {
  margin: 0 2em;
  padding: 0 1em;
  border: 1px dotted blue;
}
```

Con esto se obtendrá una composición similar a la que se puede ver en la figura 3, en la que los elementos con estilo se dividen en múltiples líneas.

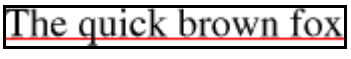
Figura 3



Los márgenes, la separación y el borde no se aplican donde se produce la rotura.

La alineación vertical de las cajas en línea dentro de la caja de línea que las contiene está determinada por la propiedad `vertical-align` (alineación-vertical). El valor por defecto es `baseline`, lo que significa que las cajas en línea se alinean de manera que sus líneas base de texto quedan alineadas. La línea base es la línea imaginaria sobre la que se sitúan las letras sin astas descendentes. Esta línea se sitúa un poco por encima de la parte inferior de la caja de línea para dejar espacio para las astas ascendentes de las letras en minúsculas, tal como muestra la figura 4.

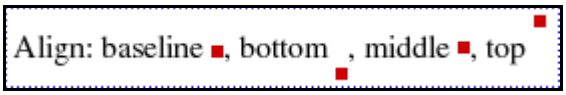
Figura 4



Las letras se sitúan sobre la línea base imaginaria.

Observad que la propiedad `vertical-align` se aplica sólo a las cajas en línea y a las celdas de tabla, y no se hereda. La figura 5 muestra algunas imágenes pequeñas con diferentes alineaciones verticales.

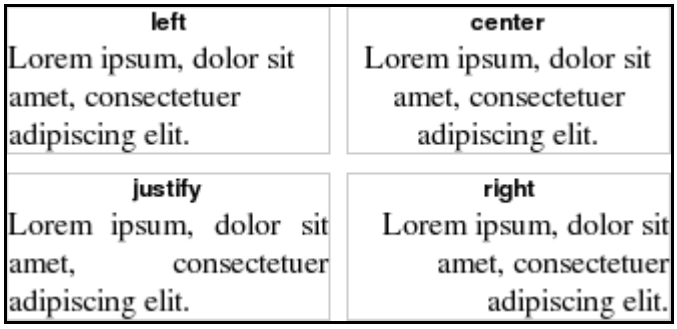
Figura 5



Las imágenes situadas con los valores de la propiedad `vertical-align` del CSS

Cuando la anchura total de las cajas en línea en una caja de línea es inferior a la anchura de la caja de línea en sí, la alineación vertical se controla con la propiedad `text-align`. Con `text-align:justify` (alinear texto: justificar) se inserta un espacio adicional entre las cajas en línea, si es necesario, para alinear el contenido a la izquierda y a la derecha. Esta propiedad se aplica a las cajas de bloque, a las celdas de tablas y a los bloques en línea, y se hereda. La figura 6 muestra el resultado de aplicar valores diferentes de la propiedad `text-align` al texto que hay en el interior de las celdas de una tabla.

Figura 6



Controlar la alineación del texto con la propiedad `text-align`

10.3. Posicionamiento relativo

El posicionamiento relativo es un sistema de posicionamiento de CSS, pero está más relacionado con el "posicionamiento" estático que con sus primos-hermanos: el posicionamiento absoluto y el fijo.

Un elemento con `position:relative` se coloca en principio igual que cualquier elemento estático; de bloque o insertado. Pero entonces sucede algo muy interesante: la caja generada se desplaza según las propiedades `top`, `bottom`, `left` y `right`.

Lo que hay que recordar sobre el posicionamiento relativo es que sólo se desplaza la caja generada. El elemento sigue estando allí donde estaba en el flujo del documento estático. Aquí es donde "ocupa espacio" respecto a los otros elementos. Eso significa que la caja desplazada puede acabar encima de otras cajas de elementos, ya que éstas siguen actuando como si el elemento con un posicionamiento relativo se hubiera quedado donde debía estar antes de aplicar el posicionamiento. Respecto al flujo del documento, el elemento no se ha movido; es sólo el resultado visual final lo que muestra la caja desplazada. Veamos cómo funciona en la práctica.

1. Copiad el siguiente código HTML en un documento nuevo del editor de textos que más os guste y guardadlo con el nombre `relative.html`.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8”>
    <title>Relative Positioning</title>
  </head>
  <body>
    <p>Lorem ipsum dolor sit amet consectetuer adipiscing elit.
    Curabitur feugiat feugiat purus.
    Aenean eu metus. Nulla facilisi.
    Pellentesque quis justo vel massa suscipit sagittis.
    Class aptent taciti sociosqu ad litora torquent per conubia nostra, per
    inceptos hymenaeos.
    Quisque mollis, justo vel rhoncus aliquam, urna tortor varius lacus, ut
    tincidunt metus arcu vel lorem.
    Praesent metus orci, adipiscing eget, fermentum ut, pellentesque non, dui.
    Sed sagittis, <span>metus a semper</span> dictum, sem libero sagittis nunc,
    vitae adipiscing leo neque vitae tellus.
    Duis quis orci quis nisl nonummy dapibus.
    Etiam ante. Phasellus imperdiet arcu at odio.
    In hac habitasse platea dictumst. Aenean metus.
    Quisque a nibh. Morbi mattis ullamcorper ipsum.
    Nullam odio urna, feugiat sed, bibendum sed, vulputate in, magna.
    Nulla tortor justo, convallis iaculis, porta condimentum, interdum nec, arcu.
    Proin lectus purus, vehicula et, cursus ut, nonummy et, diam.</p>
  </body>
</html>
```

2. Abrid el archivo con vuestro navegador web para ver qué aspecto tiene en este momento; deberíais ver sólo un párrafo de texto normal.

3. Cread un documento nuevo en vuestro editor, copiad el código CSS siguiente y guardad el archivo con el nombre `style.css`.

```
p {
  width: 20em;
}
span {
  background-color: lime;
}
```

4. Enlazad la hoja de estilo en el documento HTML insertando la línea siguiente justo antes de la etiqueta `</head>`.

```
<link rel="stylesheet" type="text/css" href="style.css">
```

5. Guardad los dos archivos y actualizad la página en el navegador. Hemos conseguido que el párrafo sea más estrecho para que los saltos de línea estén siempre en la misma posición incluso cuando la ventana del navegador sea pequeña. Ahora el elemento `span` tiene un color de fondo un poco chillón para hacerlo más visible.

6. A continuación, modificaremos la hoja de estilos añadiendo tres declaraciones a la regla para el elemento `span`:

```
span {
  position: relative;
  top: 1em;
  left: 2em;
  background-color: lime;
}
```

7. Guardad y volved a cargar la página en el navegador para ver los efectos del posicionamiento relativo.

Habéis desplazado el elemento `span` tanto vertical como horizontalmente. Observad que queda superpuesto sobre la línea de texto siguiente y que en el lugar donde se encontraba ahora hay un espacio vacío.

La manera como se ha desplazado la caja generada quizá no era la que esperabais de este código. Habéis especificado `top:1em`, pero la caja se ha desplazado hacia abajo. Además, la caja también se ha desplazado hacia la derecha a pesar de haber especificado `left:2em`. ¿Por qué?

La clave para entender el funcionamiento de estas propiedades con el posicionamiento relativo es darse cuenta de que especifican el borde en el que se aplica el movimiento, y no la dirección del movimiento. Es decir, que la propiedad `top` desplaza la caja en relación con su borde superior, la propiedad `left` lo desplaza en relación con su borde izquierdo, y así sucesivamente. La caja se aleja del borde especificado; por lo tanto, `top:1em` desplaza el cuadro 1 em desde la posición superior, es decir, hacia abajo. Los números negativos desplazan el cuadro en la dirección opuesta y, por lo tanto, `bottom:-1em` es lo mismo que `top:1em`.

Eso nos lleva a otra conclusión: no tiene ningún sentido especificar al mismo tiempo una propiedad `top` y una propiedad `bottom` (o `left` y `right`) para el mismo elemento. Las reglas del CSS dicen que si se especifica `top`, entonces

`bottom` se debe ignorar. Respecto al movimiento horizontal, esto depende de la propiedad `direction`. Si se especifican `left` y `right` al mismo tiempo, en un entorno de izquierda a derecha se ignora `right` y en un entorno de derecha a izquierda se ignora `left`.

El ejemplo que hemos visto explica el posicionamiento relativo, pero no parece muy útil, ¿verdad? Así pues, ¿para qué sirve el posicionamiento relativo? Echemos un vistazo a un ejemplo más complicado.

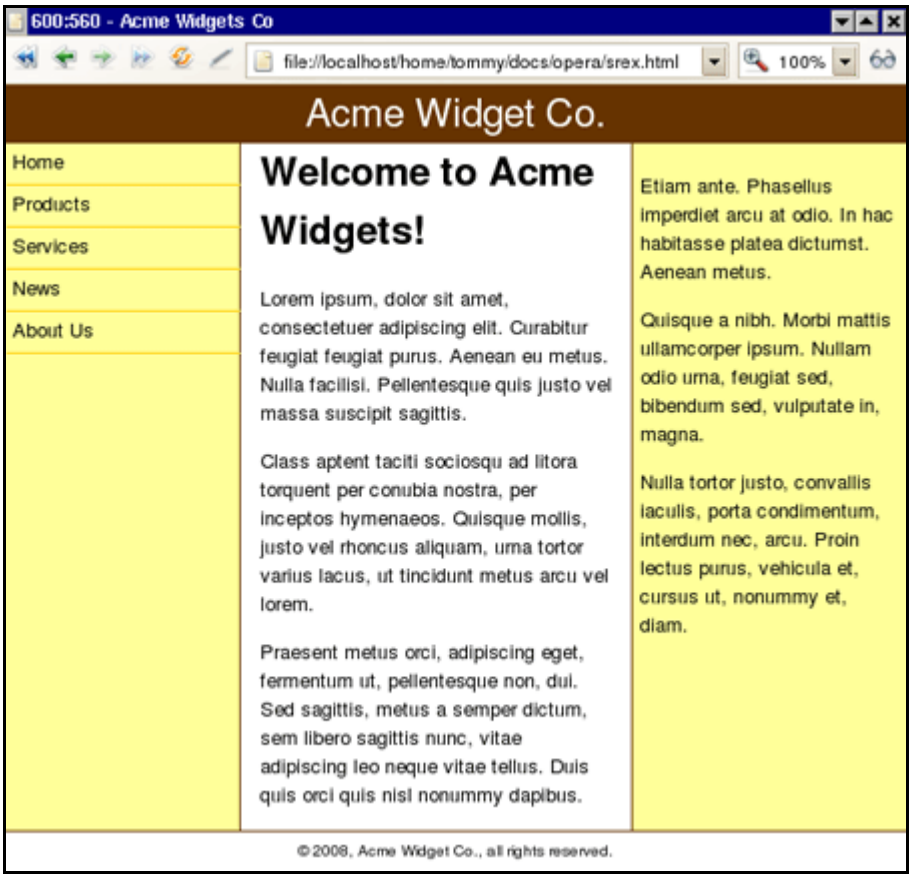
10.3.1. Disposición de múltiples columnas con requisitos de orden en el código fuente

Podéis ver el apartado 9 de este módulo.

Aviso: este ejemplo es un tanto complejo. Si es la primera vez que entráis en el mundo del CSS, os puede parecer incluso algo descorazonador, pero lo iremos explicando a un ritmo muy pausado dejando bien claro qué es lo que hacemos en cada momento. Si todavía no habéis leído el apartado anterior, que habla de los elementos flotantes y del *clearing*, ahora sería un buen momento para hacerlo.

Hay un tipo de composición que es muy habitual en las páginas web. Incluye un encabezamiento, que normalmente contiene algún logotipo o insignia, bajo la que hay dos o más "columnas" una al lado de la otra. Al final suele haber un pie de página que ocupa toda la anchura, quizá con el aviso de copyright o la información de contacto. La figura 7 muestra un ejemplo de este tipo de composición.

Figura 7



Una composición típica con múltiples columnas entre un encabezamiento y un pie de página

En la Edad Oscura (la década de 1990), este tipo de distribución se solía crear con tablas. Esto es un uso inadecuado del etiquetado del HTML para finalidades presentacionales, nada aconsejable; por lo tanto, en esta asignatura no lo enseñaremos. CSS ofrece maneras de conseguir esto mismo con `display: table-cell` y similares, pero esta solución tiene un inconveniente importante: no hay ninguna versión de Internet Explorer que lo acepte, de modo que tampoco la veremos. Sólo nos quedan dos opciones: los elementos flotantes y el posicionamiento absoluto. Estos dos métodos tienen ventajas e inconvenientes, pero si queréis un pie de página que ocupe toda la anchura y no sabéis de entrada qué columna será la más larga, entonces necesitaréis los elementos flotantes para garantizar la integridad del diseño.

El problema de los elementos flotantes es que sólo se desplazan hacia la derecha o la izquierda hasta que tocan el borde del bloque padre u otro elemento flotante. Esto quiere decir que las columnas flotantes deben aparecer en el orden correcto en el etiquetado. Pero algunas veces se quiere tener un orden presentacional diferente del orden del código fuente. Quizá os interese que el contenido aparezca antes de la navegación, por ejemplo, con el fin de mejorar la usabilidad de la navegación con el teclado y la optimización en motores de búsqueda. Esto se puede conseguir, incluso con elementos flotantes, haciendo un buen uso de los márgenes negativos y el posicionamiento relativo; veamos cómo hacerlo. Empezaremos con un documento HTML que nos servirá de base para ir trabajando.

1. Copiad el código siguiente en vuestro editor de textos y guardad el archivo con el nombre `layout.html`.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset=utf-8">
    <title>Static and Relative Positioning</title>
    <link rel="stylesheet" type="text/css" href="layout.css">
  </head>
  <body>
    <div id="header">Header</div>
    <div id="main">Main content</div>
    <div id="sidebar">Sidebar</div>
    <div id="nav">Navigation</div>
    <div id="footer">Footer</div>
  </body>
</html>
```

2. A continuación, crearemos el embrión de una hoja de estilos. Copiad el código siguiente en vuestro editor de textos y guardad el archivo con el nombre `layout.css`.

```
#header {
  background-color: #369;
  color: #fff;
}
#sidebar {
  background-color: #ff6;
}
#nav {
  background-color: #ddd;
}
#footer {
  border-top: 1px solid #369;
}
```

3. Guardad los dos archivos y cargad la página en el navegador. Las cinco divisiones aparecen en orden, de arriba abajo.

Imaginaos que el departamento de diseño ha especificado que la navegación debe ir a la izquierda, la barra lateral a la derecha y el contenido principal en la columna de en medio. El encabezamiento y el pie de página deben ocupar toda la anchura de la página, pero no sabemos cuál de las tres columnas será la más larga. El orden del código fuente viene determinado por los expertos en accesibilidad y usabilidad y no es negociable. ¿Cómo se pueden combinar todos estos requisitos para conseguir una composición que funcione?

Para que funcione, deberéis añadir un elemento adicional al etiquetado. Es inevitable, pero un elemento adicional es algo que no os debería preocupar demasiado. Necesitaréis un elemento que envuelva las tres "columnas".

4. Insertad las dos líneas destacadas en el documento HTML:

```
<div id="header">Header</div>
  <div id="wrapper">
    <div id="main">Main content</div>
    <div id="sidebar">Sidebar</div>
    <div id="nav">Navigation</div>
  </div>
<div id="footer">Footer</div>
```

Los diseñadores (que, por suerte, son conscientes de la accesibilidad y de la independencia de dispositivo) han estipulado que la navegación debe tener una

anchura de 12 em y la barra lateral, de 14 em. La columna con el contenido principal debe tener una anchura fluida, de manera que la composición se adapte a diferentes tamaños de ventanas, ya que las composiciones con una anchura fija no son demasiado fáciles de utilizar por el usuario. Para evitar que las líneas de texto sean demasiado largas y dificulten la legibilidad, deberéis limitar la composición a una anchura máxima. Para evitar el solapamiento en ventanas sumamente estrechas, también deberéis limitar la composición a una anchura mínima. Dentro de estas limitaciones, la composición se debe centrar horizontalmente en la ventana del navegador.

5. A continuación, asignad las anchuras a la navegación y a la barra lateral y definid las limitaciones de anchura y el centrado general añadiendo las reglas siguientes al final del archivo CSS:

```
body{
  margin: 0 auto;
  min-width: 40em;
  max-width: 56em;
}
#sidebar {
  width: 13em;
  padding: 0 0.5em;
  background-color: #ff6;
}
#nav {
  width: 11em;
  padding: 0 0.5em;
  background-color: #ddd;
}
```

6. Guardad los archivos y volved a cargarlos; deberíais ver la barra lateral amarilla y la navegación de color gris con las anchuras deseadas. Si la ventana del navegador es bastante ancha, también veréis que toda la página tiene una anchura limitada y que aparece centrada horizontalmente.

7. Intentad cambiar el tamaño de la ventana y mirad cómo se adapta la composición.

> **Nota**
>
> Si seguís los ejemplos con versiones de Internet Explorer hasta la 8 veréis resultados extraños porque Internet Explorer tenía muchos errores de representación. En este ejemplo nos centraremos en la manera de hacer las cosas según los estándares.

Si observáis el código con atención, veréis que las anchuras están fijadas en 13 em y 11 em en lugar de 14 em y 12 em. Esto es así porque necesitamos un poco de separación horizontal; no queremos que el contenido de estas columnas toque los bordes, ya que no queda demasiado bien. La separación se añade a la anchura, con lo cual 13 em + 0,5 em + 0,5 em suman un total de 14 em, que es lo que queremos.

Crear columnas

Muy bien, ahora ya tenemos los componentes básicos, pero se muestran uno después de otro. Como queremos tres columnas, las deberemos empezar a hacer flotar.

1. Añadid las reglas siguientes a vuestro archivo CSS:

```
#main {
  float: left;
}
#sidebar {
  float: left;
  width: 13em;
  padding: 0 0.5em;
  background-color: #ff6;
}
#nav {
  float: left;
  width: 11em;
  padding: 0 0.5em;
  background-color: #ddd;
}
```

Con este código se hacen flotar las columnas, sí, pero no se muestran en el orden correcto. Además, la columna del contenido principal es demasiado estrecha. ¿Y qué ha pasado con el pie de página?

2. En primer lugar, nos encargaremos del pie de página. El problema es que hemos hecho flotar las tres columnas, con lo que han quedado fuera del flujo del documento. El pie de página queda justo debajo del encabezamiento y la caja de línea que contiene el texto se ha acortado, de manera que la palabra "Footer" aparece a la derecha de los elementos flotantes. Podemos solucionarlo haciendo que el pie de página esté a una cierta distancia de todas las columnas flotantes. Añadid la regla siguiente a vuestro archivo CSS:

```
#footer {
  clear: left;
  border-top: 1px solid #369;
}
```

3. Y ahora nos dedicaremos a las tres columnas. Lo haremos paso a paso, y durante un momento todo tendrá un aspecto horrible; pero no os desesperéis porque al final todo acabará bien.

La clave es el elemento de envoltorio. Definiremos unos márgenes izquierdo y derecho que se correspondan con las anchuras de las columnas laterales (la navegación y la barra lateral). La columna del contenido principal ocupará toda la anchura del envoltorio y las columnas laterales se desplazarán hacia el espacio que dejan vacío los márgenes. ¿Suena complicado? No os preocupéis, lo iremos haciendo poco a poco. En primer lugar definiremos los márgenes del envoltorio añadiendo la regla siguiente al archivo CSS:

```
#wrapper {
  margin: 0 14em 0 12em;
  padding: 0 1em;
}
```

Recordad que los valores de la propiedad `margin` abreviada se especifican en el orden **TRaBaLenguas**, es decir: `top` (superior), `right` (derecho), `bottom` (inferior), `left` (izquierdo). Definimos los márgenes superior e inferior a 0, el margen derecho en 14 em (para la barra lateral) y el margen izquierdo en 12 em (para la navegación). También hemos añadido 1 em de separación horizontal porque no queremos que el contenido toque las columnas laterales, ya que debe respirar.

4. El siguiente paso es hacer que la columna del contenido principal ocupe toda la anchura de su elemento envoltorio padre; el código también define un color de fondo chillón para esta columna, temporalmente:

```
#main {
  float: left;
  width: 100%;
  background-color: lime;
}
```

5. Guardad el archivo y volved a cargarlo; veréis una columna de contenido de color verde lima con la barra lateral y la navegación debajo. También veréis que hay mucho espacio en blanco a ambos lados. Lo que debemos hacer ahora es conseguir que nuestras columnas laterales se desplacen hacia este espacio en blanco.

De momento, nos dedicaremos a la barra lateral, que es flotante y ya tiene la anchura correcta, pero como la columna `#main` tiene una anchura del 100%

la barra lateral se desplaza hacia abajo. ¿Cómo podemos hacer que suba y se coloque al lado de `#main` teniendo en cuenta que `#main` ocupa toda la anchura? Lo haremos en dos pasos: en primer lugar, la moveremos hacia arriba y, después, la desplazaremos hacia el margen.

6. Para conseguir que la barra lateral flotante, que se ha desplazado hacia abajo, vuelva a ir hacia arriba, utilizaremos un truco muy ingenioso. Añadid lo siguiente a la regla `#sidebar`:

```
#sidebar {
  float: left;
  width: 13em;
  padding: 0 0.5em;
  background-color: #ff6;
  margin-left: -14em;
}
```

7. Guardad, volved a cargar y veréis que la barra lateral se encuentra ahora a la misma altura vertical que la columna del contenido. Con un margen izquierdo negativo igual a la anchura de la barra lateral, estamos moviendo el elemento hacia el interior del envoltorio y ya no se desplaza hacia abajo. El problema es que queda sobre el contenido.

8. Deberéis desplazarla hacia el margen sin que vuelva a ir hacia abajo, y aquí es donde entra en juego finalmente el posicionamiento relativo. Éste hace precisamente lo que queremos: desplaza la caja generada sin mover el elemento en sí. Añadid las propiedades destacadas del código siguiente a la regla para `#sidebar`:

```
#sidebar {
  float: left;
  width: 13em;
  padding: 0 0.5em;
  background-color: #ff6;
  margin-left: -14em;
  position: relative;
  left: 15em;
}
```

Observad que hemos tenido que desplazar el elemento 15 em, y no 14 em. Esto es así porque en el envoltorio hay 1 em de separación a la derecha que es necesario que superemos. La barra lateral ya se encuentra en el lugar donde debe estar: hacia el margen, al lado de la columna del contenido y bien alineada con los bordes derechos del encabezamiento y el pie de página.

9. Ahora debéis hacer lo mismo con la navegación; el procedimiento es similar, pero hay un pequeño detalle que debéis tener en cuenta. Mover y desplazar la barra lateral ha sido fácil porque los movimientos eran básicamente los mismos que la anchura de la columna: un margen negativo de 14 em y un desplazamiento de 14 em + 1 em hacia la derecha. Pero la columna de la navegación se debe mover por encima de la columna del contenido y entonces todavía se debe desplazar más hacia el margen.

Aquí nuestros aliados serán los porcentajes. Un valor de porcentaje en los márgenes de la columna de la navegación será relativo a la anchura de su padre, el envoltorio. Como queréis mover la columna hasta el extremo del envoltorio, añadid la propiedad destacada del código siguiente a la regla para `#nav`:

```
#nav {
  float: left;
  width: 11em;
  padding: 0 0.5em;
  background-color: #ddd;
  margin-left: -100%;
}
```

10. ¡Ya lo tenemos! Guardad, volved a cargar y veréis la navegación sobre la parte izquierda de la columna del contenido. Todo lo que debéis hacer es desplazarla hacia el margen. Añadid las siguientes propiedades resaltadas a la regla para `#nav`:

```
#nav {
  float: left;
  width: 11em;
  padding: 0 0.5em;
  background-color: #ddd;
  margin-left: -100%;
  position: relative;
  right: 13em;
}
```

Aquí también la anchura de la navegación es de 12 em, pero tenemos 1 em de separación del envoltorio que debemos superar, por lo que es necesario que desplacemos la caja 13 em. Lo estáis desplazando hacia la izquierda, es decir, desde el borde derecho, y por ello utilizamos la propiedad `right`.

11. Eliminad el fondo de color verde lima de la columna del contenido y ya lo tendréis todo.

Solucionar los problemas con Internet Explorer

Hay dos propiedades de esta composición que hacen que no funcione bien con Internet Explorer 6 para Windows. Uno de los problemas es que IE 6 no acepta las propiedades `min-width` (anchura mín.) y `max-width` (anchura máx.) y el otro es que IE es muy malo con los porcentajes.

Para emular las restricciones de anchura, podéis utilizar la notación `expression()` exclusiva de Microsoft. Toma una expresión de JScript como argumento y proporciona el valor de retorno de esta expresión. Si exige hacer muchos cálculos, esta expresión puede provocar problemas de rendimiento, ya que se calcula cada vez que el navegador debe definir la anchura de `body`. También exige que JScript esté activado, pero podéis añadir una degradación muy elegante, de manera que si, por ejemplo, JScript no está disponible, el diseño recurrirá a una alternativa que seguirá siendo utilizable. En este ejemplo haréis que la maquetación sea totalmente elástica en lugar del diseño fluido limitado que hemos creado antes si JScript está desactivado.

La manera recomendada de utilizar reglas de estilos para solucionar problemas en Internet Explorer es utilizando los "comentarios condicionales". Ésta es una función única de Microsoft que integra lógica condicional en los comentarios del HTML (hay un apartado dedicado exclusivamente a los comentarios condicionales* en dev.opera.com).

*** http://dev.opera.com/articles/view/supporting-ie-with-conditional-comments/**

1. Añadid las líneas siguientes en el código HTML justo antes de la etiqueta `</head>`:

```
<!--[if lte IE 6]>
  <link rel="stylesheet" type="text/css" href="layout-ie6.css">
<![endif]-->
```

2. Ahora cread un archivo nuevo con el nombre `layout-ie6.css` que contenga lo siguiente:

```
body{
  width: 50em;
  width: expression(w=document.documentElement.offsetWidth,
 em=document.getElementById("nav").offsetWidth/12,
  (w<40*em?"40em":(w>56*em?"56em":"auto")));
}
#wrapper {
  height: 1em;
}
#nav {
  margin-left: -22em;
```

```
  margin-left: expression((-(document.getElementById("wrapper").clientWidth))
  +"px");
  left: 13em;
}
```

Así se solucionan los dos problemas de IE 6. Utilizamos expresiones JScript para emular las propiedades `min-width` y `max-width`, que IE 6 no acepta, con un valor alternativo elástico de 50 em. A continuación utilizamos otra expresión JScript para definir un margen izquierdo en píxeles en lugar de porcentajes, también con un valor alternativo elástico. La altura para `#wrapper` es sólo para habilitar la propiedad `hasLayout` específica de Microsoft, que necesitamos para que el posicionamiento relativo de la navegación funcione correctamente. Microsoft ha documentado `hasLayout`* en la MSDN, pero no es precisamente fácil de entender.

*** http://msdn.microsoft.com/library/shared/deeptree/asp/rightframe.asp?dtcfg=/library/deeptreeconfig.xml&url=/library/en-us/IETechCol/cols/dnexpie/expie20050831.asp?frame=true&hidetoc=false**

¿Y qué sucede con IE 7? Sí acepta `min-width` y `max-width`, pero sigue situando mal el elemento de la navegación; es el mismo problema de `hasLayout` de IE 6 que vuelve a asomar la cabeza. Debéis habilitar `hasLayout` en el elemento `#wrapper`. Por suerte, podéis hacerlo de manera que no comprometa a los navegadores conformes con los estándares, o sea que no será necesario que creéis una hoja de estilos diferente para IE 7; podéis añadir sencillamente la regla siguiente para manipular el envoltorio:

```
#wrapper {
  margin: 0 14em 0 12em;
  padding: 0 1em;
  min-height: 1em;
}
```

La definición de una altura mínima habilita `hasLayout` y no provoca ningún problema en otros navegadores, es decir, lo podéis poner en la hoja de estilos principal.

Estas soluciones no son perfectas; si el tamaño de la ventana se modifica en determinadas dimensiones, la composición seguirá haciendo cosas raras en IE 6 e IE 7, aunque si se vuelve a cargar la página, la composición quedará bien otra vez.

10.3.2. Otros usos del posicionamiento relativo

El uso más habitual del posicionamiento relativo no implica desplazar la caja generada. Esto puede sonar extraño: ¿por qué deberíais utilizar el posiciona-

miento relativo sin desplazar la caja? Explicaremos esta razón en el apartado siguiente porque también tiene que ver con el posicionamiento absoluto. Así pues, deberéis esperar un poco.

Definir `position:relative` (sin desplazar la caja) también puede ser útil con algunos problemas de representación extraños de Internet Explorer. Activa la famosa propiedad interna `hasLayout`, que tiene un impacto muy importante en la representación que hace Internet Explorer de los elementos.

Resumen

El posicionamiento estático es la manera por defecto de hacer las cosas. Las cajas de bloque se distribuyen verticalmente según el orden en el que aparecen en el código fuente, mientras que las cajas en línea se distribuyen horizontalmente en cajas en línea dentro de estas cajas de bloque.

El posicionamiento relativo permite desplazar la caja generada en una o dos dimensiones. El elemento sigue ocupando espacio como si fuera estático, pero la caja generada se puede desplazar a otra posición. El posicionamiento relativo se utiliza principalmente en combinación con elementos flotantes para crear composiciones en las que el orden de la presentación es diferente del orden del código fuente.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿Qué sucede cuando dos márgenes adyacentes de un contexto de formato estático se superponen y uno o ambos márgenes son negativos?

2. Añadid un borde vertical entre cada una de las columnas laterales y la columna del contenido principal. Recordad que las tres columnas son flotantes, con lo cual la altura del elemento de envoltorio ha pasado a ser cero.

3. ¿Cómo podéis hacer que todas las columnas tengan la misma altura (o que como mínimo lo parezca), de manera que los colores de fondo lleguen hasta el pie de página sea cual sea la columna más larga? (Consejo: buscad “faux columns” en vuestro motor de búsqueda preferido.)

11. Posicionamiento absoluto y fijo con CSS

Tommy Olsson

Este es el momento de fijarnos en el segundo par de valores de propiedad de `position`: `absolute` (absoluto) y `fixed` (fijo). El primer par de valores: `static` (estática) y `relative` (relativo) están íntimamente relacionados, y ya los vimos en el apartado anterior.

Los elementos posicionados absolutamente se eliminan completamente del flujo del documento. Esto significa que no tienen ningún efecto sobre su elemento padre ni sobre los elementos que aparecen después de ellos en el código fuente. Por lo tanto, un elemento posicionado absolutamente se superpondrá sobre otros contenidos a no ser que se tome alguna medida para evitarlo. En ocasiones, por supuesto, esta superposición es exactamente lo que queréis, pero deberíais ser conscientes de ello para aseguraros de que obtenéis la composición que queréis.

El posicionamiento fijo es de hecho una forma especializada del posicionamiento absoluto; los elementos con posicionamiento fijo están fijos en relación con la ventana o *viewport* del navegador en vez de estarlo en relación el elemento continente; incluso si se desplaza la página, se mantienen exactamente en la misma posición en la ventana del navegador.

En este apartado os daremos algunos ejemplos prácticos del uso de los posicionamientos `absolute` y `fixed`, observaremos algunos pequeños problemas de compatibilidad de navegadores y nos fijaremos en el concepto de índice *z*.

Pero antes de empezar a hablar de todo esto, trataremos un concepto previo esencial: los bloques contenedores.

11.1. Bloques contenedores

Un concepto fundamental respecto al posicionamiento absoluto es el bloque contenedor: la caja de bloque respecto a la que se encuentran la posición y las dimensiones de la caja posicionada absolutamente.

Para las cajas estáticas y posicionadas relativamente, la caja contenedora es la antepasada del bloque más próximo o, dicho de otro modo, el elemento padre. No obstante, para los elementos posicionados absolutamente es algo más complicado. En este caso, la caja contenedora es el antepasado posicionado más próximo. Con *posicionado* nos referimos a un elemento cuya propiedad `position` está establecida en `relative`, `absolute` o `fixed`, es decir, cualquier cosa excepto elementos estáticos normales.

De manera que, establecer `position:relative` para un elemento lo convierte en bloque contenedor de cualquier descendiente posicionado absolutamente (elementos hijos), tanto si aparecen inmediatamente debajo del elemento posicionado relativamente en la jerarquía, o más abajo.

Si un elemento posicionado absolutamente no tiene un antepasado posicionado, entonces el bloque contenedor es lo que se llama *bloque contenedor inicial,* que en la práctica equivale al elemento `html`. Si estáis mirando la página web en pantalla, se refiere a la ventana del navegador; si vais a imprimir la página, se refiere a los límites de la página.

Los elementos con posicionamiento fijo difieren ligeramente, ya que su bloque contenedor siempre es el bloque contenedor inicial.

Así pues, resumimos en una serie de sencillos pasos; para encontrar el bloque contenedor de un elemento con `position:absolute`, esto es lo que debéis hacer:

1. Mirad el elemento padre del elemento posicionado absolutamente: ¿la propiedad `position` de este elemento tiene uno de los valores `relative`, `absolute` o `fixed`?

2. Si es así, habéis encontrado el bloque contenedor.

3. En caso contrario, pasad al elemento padre del padre y empezad de nuevo desde el paso 1 hasta que encontréis el bloque contenedor u os quedéis sin antepasados.

4. Si habéis llegado al elemento `html` sin encontrar un antepasado posicionado, entonces el bloque contenedor es el elemento `html`.

11.2. Posicionamiento absoluto

El posicionamiento fijo es una forma especial de posicionamiento absoluto, de manera que lo estudiaremos más adelante y ahora nos concentraremos en el caso más generalizado. A no ser que se indique lo contrario, cuando utilizamos el término *posicionado absolutamente* desde ahora hasta el final del apartado, nos estaremos refiriendo tanto a elementos con `position:fixed` como a elementos con `position:absolute`.

11.2.1. Especificación de la posición

Con el posicionamiento relativo, habéis aprendido que las propiedades `top`, `right`, `bottom` y `left` se pueden utilizar para especificar la posición de la caja. Para

especificar la posición de una caja posicionada absolutamente se utilizan las mismas propiedades, pero la manera de utilizarlas es bastante diferente.

Para un elemento posicionado relativamente, las cuatro propiedades especifican la distancia relativa para desplazar la caja generada. Recordad que en el caso del posicionamiento relativo se complementan entre sí, de manera que `top:1em` y `bottom:-1em` quieren decir lo mismo, y no tiene mucho sentido especificar tanto `top` como `bottom` (o `left` y `right`) para el mismo elemento porque uno de los valores se ignorará.

Estos puntos no son ciertos en el caso del posicionamiento absoluto. En este caso, se pueden utilizar las cuatro propiedades al mismo tiempo para especificar la distancia desde cada borde del elemento posicionado hasta el borde correspondiente del bloque continente. También se puede especificar la posición de una de las esquinas del cuadro posicionado absolutamente, por ejemplo utilizando `top` y `left`, y entonces especificar las dimensiones del cuadro mediante `width` y `height` (o simplemente no utilizar `width` y `height` si queréis dejar que se ajuste para encajar su contenido).

La versión 6 de Microsoft Internet Explorer (y anteriores) no acepta el método de especificar los cuatro bordes, pero sí el método de especificar una esquina más las dimensiones.

```
/* Este método funciona en IE6 */
#foo {
  position: absolute;
  top: 3em;
  left: 0;
  width: 30em;
  height: 20em;
}
/* Este método no funciona en IE6 */
#foo {
  position: absolute;
  top: 3em;
  right: 0;
  bottom: 3em;
  left: 0;
}
```

Lo que debéis recordar en este caso es que los valores que habéis establecido para las propiedades `top`, `right`, `bottom` y `left` especifican las distancias desde los bordes del elemento hasta los bordes de su bloque continente correspondiente. No es como en un sistema de coordenadas donde cada valor es relativo a un punto de origen. Por ejemplo, `right:2em` significa que el borde

derecho del cuadro posicionado absolutamente estará a 2 em del borde derecho del bloque contenedor.

Es absolutamente esencial saber cuál es el bloque contenedor cuando se utiliza el posicionamiento absoluto. Por ello es tan útil configurar `position:relative` en el bloque contenedor, incluso si no se desplaza realmente la posición del cuadro. Esto permite hacer que un elemento sea el bloque contenedor de sus descendientes posicionados absolutamente, es decir, os proporciona el control.

Probemos un ejemplo para ver cómo funciona.

1. Copiad el código siguiente en vuestro editor de texto y guardad el documento como `absolute.html`.

```
<!DOCTYPE html>
<html>
  <head>
   <meta charset="utf-8">
   <title>Absolute Positioning</title>
   <link rel="stylesheet" type="text/css" href="absolute.css">
  </head>
  <body>
    <div id="outer">
      <div id="inner"></div>
    </div>
  </body>
</html>
```

2. A continuación, copiad el código siguiente en un nuevo archivo y guardadlo como `absolute.css`.

```
html, body {
  margin: 0;
  padding: 0;
}
#outer {
  margin: 5em;
  border: 1px solid #f00;
}
#inner {
  width: 6em;
  height: 4em;
  background-color: #999;
}
```

3. Guardad los dos archivos y cargad el documento HTML en vuestro navegador. Veréis un rectángulo gris rodeado de un marco algo más ancho de color rojo. El elemento `#inner` tiene una anchura y altura especificadas y un color gris de fondo. El elemento `#outer`, que es el padre estructural de `#inner`, tiene un marco rojo. También tiene un margen de 5 em alrededor para alejarlo de los bordes de la ventana del navegador y dejarnos ver más claramente qué es lo que ocurre.

Nada sorprendente hasta ahora, ¿verdad? La altura del elemento `#outer` viene dada por su elemento hijo (`#inner`) y la anchura por los márgenes horizontales.

4. Mirad qué sucede ahora si hacéis que el elemento `#inner` esté posicionado absolutamente. Añadid la siguiente línea resaltada en la regla de `#inner`:

```
#inner {
  width: 6em;
  height: 4em;
  background-color: #999;
  position: absolute;
}
```

5. Guardad y recargad. En vez de un marco rojo en torno al rectángulo gris, ahora vemos lo que parece un marco más grueso sólo en la parte superior. Y el cuadro gris no se ha movido en absoluto. ¿Os lo esperabais?

Pasan dos cosas interesantes aquí: en primer lugar, hacer que `#inner` esté posicionado absolutamente lo ha eliminado completamente del flujo del documento. Esto significa que su padre, `#outer`, ahora no tiene hijos que se encuentren en el flujo normal y, por lo tanto, su altura se reduce a cero. Lo que parece una línea roja de 2 píxeles de grueso es realmente un borde de 1 píxel alrededor de un elemento con altura cero: estáis viendo los bordes superiores e inferiores sin nada en medio.

Lo segundo interesante es que el cuadro posicionado absolutamente no se ha movido. El valor predeterminado para las propiedades `top`, `right`, `bottom` y `left` es `auto`, lo que significa que el cuadro posicionado absolutamente aparecerá exactamente donde habría estado si no hubiera sido posicionado. Pero como se ha eliminado del flujo, se superpondrá a cualquier elemento del flujo normal que lo siga.

Esto es realmente muy útil: podéis utilizarlo si sólo queréis mover un cuadro generado en una dimensión. Por ejemplo, en un menú desplegable de CSS, los paneles “desplegables” se pueden posicionar absolutamente especificando

sólo la propiedad `top`. Entonces aparecerán automáticamente en la coordenada prevista a lo largo del eje X (igual que su padre).

6. A continuación, establecemos una altura para el elemento `#outer` de manera que vuelva a parecer un rectángulo y movemos `#inner` lateralmente. Añadid las siguientes líneas resaltadas a las reglas de CSS:

```
#outer {
  margin: 5em;
  border: 1px solid #f00;
  height: 4em;
}
#inner {
  width: 6em;
  height: 4em;
  background-color: #999;
  position: absolute;
  left: 1em;
}
```

7. Guardad y recargad y veréis algunos cambios. El elemento `#outer` ahora es un rectángulo otra vez, ya que le habéis especificado una altura. El elemento `#inner` se ha desplazado lateralmente, pero no donde se podría esperar encontrarlo. No está a 1 em del borde izquierdo de su padre, sino a 1 em del borde izquierdo de la ventana.

El motivo es que, como se ha explicado antes, `#inner` no tiene un antepasado posicionado, de manera que su bloque continente es el bloque continente inicial. Si especificáis una posición diferente de `auto`, es relativa al borde correspondiente del bloque continente. Cuando habéis establecido `left:1em`, el borde izquierdo de `#inner` ha acabado a 1 em del borde izquierdo de la ventana.

8. Si en vez de esto lo queréis a 1 em del borde izquierdo de su elemento padre, debéis hacer padre al bloque contenedor. Para hacerlo, ahora utilizaréis el truco que hemos mencionado anteriormente en este apartado: hacer que el bloque padre esté posicionado relativamente. Añadid la siguiente línea resaltada a la regla `#outer`:

```
#outer {
  margin: 5em;
  border: 1px solid #f00;
  height: 4em;
  position: relative;
}
```

9. Guardad y recargad: ¡pasen y vean! El rectángulo gris ahora está a 1 em del borde izquierdo del elemento padre. Establecer `position:relative` en la regla `#outer` ha hecho que esté posicionado y lo ha establecido como bloque contenedor para cualquier descendiente posicionado relativamente que pueda tener. El `left:1em` que habéis establecido para `#inner` ahora cuenta a partir del borde izquierdo de `#outer`, no del borde izquierdo de la ventana del navegador.

11.2.2. Especificación de las dimensiones

Los elementos posicionados absolutamente se ajustarán para encajar su contenido a no ser que especifiquéis sus dimensiones. Podéis especificar la anchura o bien definiendo las propiedades `left` y `right`, o bien definiendo la propiedad `width`. Podéis especificar la altura definiendo las propiedades `top` y `bottom`, o definiendo la propiedad `height`.

Cualquiera de estas seis propiedades se puede especificar como porcentaje. Los porcentajes son, por su propia naturaleza, relativos a alguna otra cosa. En este caso, son relativos a las dimensiones del bloque contenedor.

Para un elemento posicionado absolutamente, los valores de porcentaje para las propiedades `left`, `right` y `width` son relativos a la anchura del bloque continente. Los valores de porcentaje para las propiedades `top`, `bottom` y `height` son relativos a la altura del bloque continente.

El navegador Internet Explorer 6 y anteriores, y también Opera 8 y anteriores, lo interpretaron de manera totalmente errónea y utilizaron las dimensiones del bloque padre en su lugar. Vamos a probar con otro ejemplo para ver cómo esto puede suponer una gran diferencia.

1. Empezad especificando las dimensiones de `#inner` mediante valores de porcentaje; haced los siguientes cambios en la regla `#inner`:

```
#inner {
  width: 50%;
  height: 50%;
  background-color: #999;
  position: absolute;
  left: 1em;
}
```

2. Guardad y recargad, y veréis que el rectángulo gris se hace más ancho y más corto (como mínimo si estáis utilizando un navegador moderno). El bloque continente es todavía `#outer` porque tiene `position:relative`. La anchura

del elemento `#inner` ahora es la mitad de la de `#outer`, y su altura es la mitad de la altura de `#outer`.

3. Hacemos que el *viewport* sea el bloque contenedor una vez más, para ver la diferencia. Haced el siguiente cambio en `#outer`:

```
#outer {
  margin: 5em;
  border: 1px solid #f00;
  height: 4em;
  position: static;
}
```

4. Guardad y recargad; una diferencia notable, ¿verdad? El cuadro gris es ahora la mitad de ancho y la mitad de alto que la ventana del navegador.

Como podéis ver, saber cuáles son vuestros bloques contenedores es absolutamente esencial.

11.2.3. La tercera dimensión: índice Z

Es natural considerar que una página web tiene dos dimensiones. La tecnología no ha evolucionado tanto como para que las pantallas 3D estén generalizadas, de manera que nos debemos conformar con anchura, altura y falsos efectos 3D. Pero la representación CSS realmente se produce en tres dimensiones. Esto no quiere decir que pueda hacer que un elemento vuele delante del monitor (aún), pero puede hacer otras cosas útiles con los elementos posicionados.

Los dos ejes principales de una página web son el eje horizontal X y el eje vertical Y. El origen de este sistema de coordenadas se encuentra en la esquina superior izquierda de la ventana, es decir, donde tanto el valor X como Y son 0.

Pero también hay un eje Z, que podemos imaginar como perpendicular a la superficie del monitor (o del papel, si se trata de una impresión). Los valores Z más altos indican una posición de "delante" de los valores Z más bajos. Los valores Z también pueden ser negativos, y en este caso indican una posición "detrás" de algún punto de referencia (explicaremos este punto de referencia a continuación).

> **Nota**
>
> Antes de continuar, debemos advertiros que éste es uno de los temas más complicados de CSS, así que no os desmoralicéis si no lo entendéis a la primera.

Los elementos posicionados (incluidos los elementos posicionados relativamente) se representan dentro de lo que se conoce como contexto de apilamiento. Los elementos dentro de un contexto de apilamiento tienen el mismo punto de referencia a lo largo del eje Z. A continuación lo explicaremos más detalladamente. Podéis cambiar la posición Z (también denominada nivel de pila) de un elemento posicionado utilizando la propiedad `z-index`. El valor puede ser un número entero (que puede ser negativo) o una de las palabras clave `auto` o `inherit`. El valor predeterminado es `auto`, lo que quiere decir que el elemento tiene el mismo nivel de pila que su padre.

Debéis tener en cuenta que sólo podéis especificar una posición índice a lo largo del eje Z. No podéis hacer que un elemento aparezca 19 píxeles detrás o 5 centímetros delante de otro. Pensad en eso como si fuera una baraja de cartas: Podéis apilar las cartas y decidir que el as de espadas esté sobre el tres de diamantes; cada carta tiene su nivel de pila, o índice Z.

Si especificáis `z-index` como un entero positivo, le asignáis un nivel de pila "delante de" el resto de elementos del mismo contexto de apilamiento que tienen un nivel de pila inferior. Un `z-index` de 0 (*zero*) significa lo mismo que `auto`, pero hay una diferencia de la que hablaremos en un momento. Un valor entero negativo para `z-index` asigna un nivel de pila "detrás de" el nivel de apilamiento padre.

Cuando dos elementos del mismo contexto de apilamiento tienen el mismo nivel de pila, lo que aparece posteriormente al código fuente aparecerá encima de sus hermanos predecesores.

De hecho, no puede haber menos de siete capas en un contexto de apilamiento, y en cada una de estas capas puede haber cualquier número de elementos, pero no os preocupéis: lo más probable es que nunca tengáis que tratar con siete capas en un contexto de apilamiento. El orden en el que los elementos (todos los elementos, no sólo los posicionados) se representan en un contexto de apilamiento, desde detrás hacia adelante, es el siguiente:

1. El fondo y los bordes de los elementos que forman el contexto de apilamiento.
2. **Descendientes posicionados con niveles de pila negativos.**
3. Descendientes de nivel de bloque en el flujo normal.
4. Descendientes flotantes.
5. Descendientes de nivel en línea en el flujo normal.
6. **Descendientes posicionados con el nivel de pila definido como auto o 0 (cero).**
7. **Descendientes posicionados con niveles de pila positivos.**

Las entradas resaltadas son los elementos con un nivel de pila que se puede cambiar mediante la propiedad `z-index`.

Todo puede ser bastante difícil de imaginar, de modo que, hagamos algunos experimentos prácticos para ilustrar el índice Z.

1. Empezad añadiendo la siguiente línea resaltada a vuestro pequeño documento de muestra:

```
<body>
  <div id="outer">
    <div id="inner"></div>
    <div id="second"></div>
  </div>
</body>
```

2. A continuación os explicaremos cómo restaurar el CSS de manera que `#outer` sea el bloque contenedor y cómo establecer dimensiones no porcentuales de `#inner`. Hagamos que `#outer` sea un poco más alto, también, para tener más espacio para hacer pruebas. Haced los siguientes cambios resaltados en las dos reglas:

```
#outer {
  margin: 5em;
  border: 1px solid #f00;
  height: 8em;
  position: relative;
}
#inner {
  width: 5em;
  height: 5em;
  background-color: #999;
  position: absolute;
  left: 1em;
}
```

3. Añadid una regla para el elemento `#second`, también:

```
#second {
  width: 5em;
  height: 5em;
  background-color: #00f;
  position: absolute;
  top: 1em;
  left: 2em;
}
```

4. Guardad y recargad y veréis un cuadro azul brillante que se superpone a uno gris. Ambos cuadros tienen el mismo nivel de pila (`auto`, el valor inicial, que significa un nivel de pila 0), pero el cuadro azul se representa ante el cuadro gris porque aparece más tarde en el código fuente. Podéis hacer que el cuadro gris aparezca delante asignándole un nivel de pila positivo. Sólo debéis especificar que sea mayor que 0: no hace falta exagerar y utilizar un valor como 10.000. Añadid la siguiente línea resaltada en la regla `#inner`:

```
#inner {
  width: 5em;
  height: 5em;
  background-color: #999;
  position: absolute;
  left: 1em;
  z-index: 1;
}
```

5. Guardad y recargad, y veréis cómo el cuadro gris aparece ante el cuadro azul.

Contextos de apilamiento local

El resto de este subapartado trata de los contextos de apilamiento locales. Muy probablemente, no os encontraréis con eso muy a menudo trabajando como diseñadores, a no ser que intentéis hacer algo realmente avanzado con posicionamiento absoluto, pero hemos decidido incluirlo igualmente por completitud. Si queréis, podéis saltaros esta sección.

Cada elemento que tenga el `z-index` especificado como un entero establece un contexto de apilamiento nuevo, "local", en el que el propio elemento tiene un nivel de pila 0. Ésta es la diferencia que hemos mencionado antes entre `z-index: auto` y `z-index: 0`. El anterior no establece un nuevo contexto de apilamiento, pero el último, sí.

Cuando un elemento establece un contexto de apilamiento local, los niveles de pila de sus descendientes posicionados se aplican sólo en este contexto local. Estos descendientes se pueden volver a apilar los unos respecto a los otros, y respecto a su padre, pero no respecto a los hermanos del padre. Es como si el padre formara una jaula alrededor de sus descendientes de manera que no pueden abandonarla. Los descendientes se pueden mover arriba y abajo en esta jaula, pero no pueden abandonarla. El padre y sus descendientes formarán una unidad indivisible en el contexto de apilamiento que rodea al padre.

Imaginaos que estáis clasificando el papeleo antes de entregarlo al contable que se ocupa de vuestros asuntos fiscales. Tenéis informes de gastos, recibos, confirmaciones de pedidos, etcétera. y apiláis un papel encima del otro: para

facilitarle la vida a vuestro gestor, introducís los tipos de documentos que van juntos en diferentes sobres.

Un contexto de apilamiento local es parecido a este tipo de sobre. Mantiene los elementos relacionados juntos e impide que otros elementos se mezclen entre ellos. Podéis clasificar el contenido de cada sobre como queráis, pero este orden de clasificación sólo se aplica a este sobre y no afecta a la pila de documentos en conjunto. Vuestra pila contiene ahora una mezcla de documentos independientes (elementos con el nivel de pila `auto`) y sobres (elementos con un nivel de pila entero). Los sobres con niveles de pila positivos quedan por encima de los documentos independientes, mientras que los sobres con niveles de pila negativos aparecen debajo del todo de la pila.

Cada vez que asignáis un valor entero en la propiedad `z-index` de un elemento, se crea un "sobre" que contiene este elemento y sus descendientes.

Veamos cómo funcionan estos contextos de apilamiento local. Puede parecer confuso, pero realmente no es muy diferente de todo lo que ya habéis visto. Si seguís los ejemplos, acabaréis haciéndoos una buena idea de cómo funciona todo.

1. Empezad añadiendo algún contenido a vuestros dos elementos internos; añadid las líneas resaltadas a vuestro documento HTML:

```
<div id="inner">
  <span></span>
</div>
<div id="second">
  <span></span>
</div>
```

2. Añadid una regla CSS que se aplicará a estos dos elementos `span`:

```
span {
  position: absolute;
  top: 2em;
  left: 2em;
  width: 3em;
  height: 3em;
}
```

Esto hace que los elementos `span` queden posicionados absolutamente y establece sus posiciones y dimensiones. Pero, un momento, los elementos `span` son en línea... ¿Cómo se pueden especificar las dimensiones de los elementos

en línea? La respuesta es que los elementos posicionados absolutamente, como los elementos flotantes, generan automáticamente cajas de bloque.

Las posiciones que especificáis se aplicarán con relación al bloque contenedor de cada `span`. Como ambos elementos `span` tienen un `div` posicionado absolutamente como padre, estos padres asumen la función de bloques contenedores.

3. Añadimos ahora un poco de color a los elementos `span` para que podáis ver dónde aparecen; añadid las siguientes reglas a vuestra hoja de estilos:

```
#inner span {
  background-color: #ff0;
}
#second span {
  background-color: #0ff;
}
```

4. Guardad y recargad y veréis un cuadrado amarillo en la esquina inferior derecha del cuadrado gris mayor y un cuadrado de color cian en la esquina inferior derecha del cuadrado azul mayor. Los cuadrados gris y amarillo aparecen ante los cuadrados azul y cian porque el cuadrado gris tiene un `z-index:1`.

5. ¿Y si queréis que el cuadrado cian quede por delante de todos los otros cuadrados? Todo lo que debéis hacer es aplicarle un nivel de pila más alto que al cuadrado gris. De hecho, basta con darle exactamente el mismo nivel de pila que el cuadrado gris, ya que el cuadrado cian aparece más adelante en el etiquetado. Probémoslo; efectuad el siguiente cambio en vuestro CSS:

```
#second span {
  background-color: #0ff;
  z-index: 1;
}
```

6. Guardad y recargad. Si vuestro navegador admite correctamente la recomendación CSS, el cuadrado cian debería estar ahora delante.

El cuadrado gris tiene un `z-index:1`, que quiere decir que establece un contexto de apilamiento local. En otras palabras, habéis creado uno de estos "sobres" y habéis colocado el cuadrado gris y su cuadrado hijo de color amarillo en el interior.

¿Seguís sin verlo del todo claro? A ver si con el siguiente experimento lo acabamos de aclarar.

1. Estableced un nivel de pila alto para el cuadrado amarillo para llevarlo adelante; haced el siguiente cambio en vuestro CSS:

```
#inner span {
  background-color: #ff0;
  z-index: 4;
}
```

2. Si guardáis y recargáis, veréis... que no ha habido ningún cambio en absoluto. El nivel de pila que hemos especificado para el cuadrado amarillo se aplica en el contexto de apilamiento local establecido por el cuadrado gris, es decir: el cuadrado amarillo se encuentra en un sobre junto con su padre gris. Podríais mover el cuadrado cian hasta adelante porque su padre (el cuadrado azul) no establece un contexto de apilamiento local: tiene un `z-index:auto` implícito. El cuadrado azul es un papel independiente en la pila. Los cuadrados amarillo y cian están en realidad solos en pequeños sobres individuales (tienen un nivel de pila entero y establecen contextos de apilamiento local por su cuenta).

3. Si hacéis que el cuadrado azul establezca un contexto de apilamiento local, no podréis mover el cuadrado cian hacia adelante a no ser que también llevéis a su padre (el cuadrado azul) adelante. Probémoslo. Haced los siguientes cambios en vuestro CSS:

```
#inner {

  ...

  z-index: 2;
}
#second {

  ...

  z-index: 1;
}
#second span {

  ...

  z-index: 3;
}
```

4. Guardad y recargad. Ahora, tanto el cuadrado gris como el cuadrado azul establecen contextos de apilamiento local, de manera que se crean dos sobres. En la

parte inferior de la pila hay un sobre con el nivel de pila 1 que incluye dos sobres interiores (el cuadrado azul y el cuadrado cian). En la parte superior de la pila hay un sobre con nivel de pila 2 que incluye dos sobres interiores (el cuadrado gris y el cuadrado amarillo). En el primer sobre, el cuadrado azul tiene un nivel de pila local 0 y, por lo tanto, se muestra detrás del cuadrado cian, que tiene un nivel de pila local 3. En el segundo sobre, el cuadrado gris tiene un nivel de pila lo-cal 0, de manera que aparece detrás del cuadrado amarillo con nivel de pila local 4.

La figura 1 muestra los cuatro cuadros y los dos contextos de apilamiento local desde el lateral, a lo largo del eje Z.

Figura 1

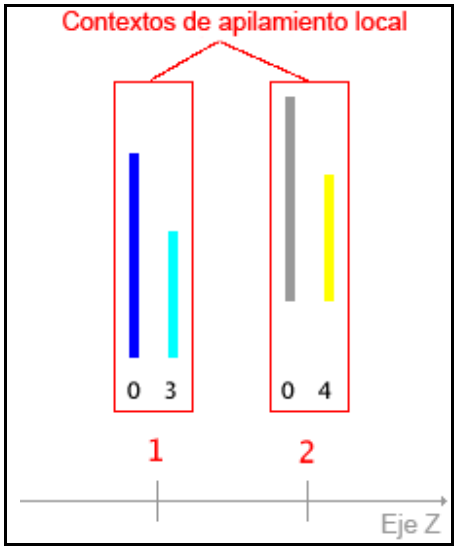


Ilustración de diferentes contextos de apilamiento. Los elementos que aparecen dentro de "2" siempre aparecerán por delante de todos los elementos dentro de "1". Después, dentro de cada contexto de apilamiento, los elementos con un número de índice z mayor aparecen por delante de los elementos con un número de índice z más pequeño. Si dos elementos tienen el mismo número de índice z, lo que aparece más tarde en el etiquetado aparecerá delante.

Esta parte probablemente os haya resultado bastante complicada, especialmente si sois nuevos con CSS. La cuestión es que debéis conocer vuestros contextos de apilamiento si estáis intentando cambiar los niveles de apilamiento de diferentes elementos. Si un elemento pertenece a un contexto de apilamiento local, sólo podéis cambiar su posición a lo largo del eje Z en este contexto local. Un elemento en un contexto de apilamiento local no se puede colar entre dos elementos de otro contexto de apilamiento local.

La buena noticia es que muy probablemente nunca os encontraréis con estos problemas. Los cambios en `z-index` no son muy comunes en las buenas composiciones, y en caso de que se produzcan, generalmente siempre es en un único contexto de apilamiento.

11.3. Posicionamiento fijo

Un elemento con `position:fixed` está fijo respecto a la ventana. Se mantiene donde está aunque se desplace el documento. Para `media="print"` se repetirá un elemento fijo en cada página impresa.

Tened en cuenta que las versiones 6 y anteriores de Internet Explorer no admiten el posicionamiento fijo de ninguna forma. Si utilizáis uno de estos navegadores, no podréis ver los resultados de los ejemplos de este subapartado.

Mientras que la posición y las dimensiones de un elemento con `position:absolute` son siempre relativas a su bloque continente, la posición y las dimensiones de un elemento con `position:fixed` son siempre relativas al bloque continente inicial. Este suele ser el *viewport*: la ventana del navegador o el cuadro de la página impresa.

Para demostrarlo, en el ejemplo siguiente haréis fijo uno de vuestros elementos. Haréis el otro muy alto para que se genere una barra de desplazamiento y así hacer más fácil el efecto que tiene.

1. Haced los siguientes cambios en vuestro código CSS:

```
#inner {
  width: 5em;
  height: 5em;
  background-color: #999;
  position: fixed;
  top: 1em;
  left: 1em;
}
#second
  width: 5em;
  height: 150em;
  background-color: #00f;
  position: absolute;
  top: 1em;
  left: 2em;
}
```

2. Guardad y recargad. Si no obtenéis una barra de desplazamiento vertical, aumentad el valor `height` para `#second` (pero ¿qué tipo de monitor gigante tenéis?).

El elemento azul alto se alarga más allá de la parte inferior de la ventana. Desplazad la página hacia abajo y observad el cuadrado gris de la esquina superior izquierda. Ahora `#inner` queda fijo en la posición a 1 em de la parte superior de la ventana y a 1 em del lado izquierdo y, por lo tanto, a medida que os desplacéis, el cuadro gris se quedará en el mismo lugar de la pantalla.

Resumen

Los elementos posicionados absolutamente se eliminan completamente del flujo del documento. Se superpondrán sobre otros contenidos a no ser que les

hagáis espacio. Si todos los elementos hijos de un contenedor están posicionados absolutamente, la altura del padre se reducirá a cero.

Los elementos posicionados absolutamente lo están respecto a un bloque contenedor, que es el antepasado posicionado más próximo. Si no hay antepasado posicionado, el *viewport* será el bloque contenedor.

Los elementos con posicionamiento fijo lo están respecto a la ventana: ésta es siempre su bloque contenedor. Siempre aparecen en el mismo lugar de la ventana del navegador cuando se ven en pantalla; cuando se imprimen, aparecen en cada página.

Las posiciones de cada borde de un elemento posicionado absolutamente se pueden especificar mediante las propiedades `top`, `right`, `bottom` y `left`. El valor de cada propiedad especifica la distancia de este borde al borde correspondiente del bloque continente del elemento.

Todos los elementos posicionados se representan en un nivel de pila determinado en un contexto de apilamiento. Podéis cambiar el nivel de pila de un elemento posicionado utilizando la propiedad `z-index`. Cuando se especifica `z-index` como un valor entero, el elemento establece un contexto de apilamiento local para sus descendientes.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. Deshaced los cambios del ejemplo de posicionamiento fijo y, a continuación, cambiad el orden de apilamiento entre los cuatro cuadrados posicionados absolutamente, de manera que el cuadrado gris esté en la parte de atrás, seguido de los cuadrados azul, amarillo y cian en este orden (consejo: eliminad todas las declaraciones de `z-index` y empezad de nuevo).

2. Moved el cuadrado amarillo hacia arriba y a la derecha especificando `top:-1em` y `left:8em`. A continuación, haced que aparezca detrás del elemento `#outer`, de manera que el borde rojo aparezca a través del cuadrado amarillo.

3. Replicad la disposición de tres columnas que creamos en el apartado de posicionamiento estático y relativo utilizando el posicionamiento absoluto en su lugar. Como será imposible tener un pie de anchura completa, podéis eliminar el elemento `#footer`, pero no podéis cambiar nada más en el etiquetado (aparte del enlace a la hoja de estilos).

4. Modificad la disposición del ejercicio anterior para hacer que la navegación utilice el posicionamiento fijo. Deberéis eliminar los márgenes horizon-

tales automáticos del elemento `body` para hacerlo posible. Añadid bastante contenido a la columna principal y/o la barra lateral para que aparezca una barra de desplazamiento, de manera que podáis verificar que funciona.