[Indice]

Escribiendo hojas de estilo

Elementos modificables con CSS

O mejor dicho, atributos de elementos que pueden modificarse con CSS. Son muchos (casi todos) los aspectos que pueden modificarse con CSS. Dado el carácter general de esta guía, aquí veremos solamente los más utilizados.

Generalmente se actúa sobre los siguientes:

- **Texto o contenido**
- **Tamaño del bloque**
- **Color del fondo del bloque**
- **Bordes de bloque**
- **Estilo de los bordes**
- **Color de los bordes**
- **Márgenes de bloque**
- **Espaciado interno del bloque**
- **Posicionamiento del bloque**
- **Gráficos y bloques flotantes**
- **Visibilidad del bloque**
- **Listas**
- **Enlaces**
- **CSS + JavaScript**
- **Efectos especiales**
- **Tablas**

Y puede que te preguntes qué es un bloque. Como **bloque** puede entenderse todo lo comprendido dentro de elementos con cierre, como <body> </body>, , <form> </form>, <table> </table>, <td> </td>, , etc., y al aplicar estilos con CSS no es buena idea escribir nada fuera de estos elementos. En lugar de ser un nombre de elemento HTML, un bloque también puede tener un nombre definido por el programador, por ejemplo "menu", y generalmente son éstos los que contienen en su interior a los anteriores o a otros bloques definidos por el usuario. El número de bloques en una página puede ser muy elevado (más de mil), y dependerá de la capacidad del navegador utilizado.

Veamos cómo funciona cada cosa:

Los **Márgenes de bloque** son el espacio comprendido entre el bloque y el borde de la ventana activa del navegador. Se controla con el atributo **margin**.

Los **Bordes de bloque**, sin aplicar estilos, no son visibles, y es como un cuadro imaginario que envuelve todo el contenido del bloque. Su atributo de control es **border**.

Espaciado interno del bloque es la distancia entre el borde del bloque y su contenido. Es el atributo **padding** (en inglés significa algo así como "acolchado")

El **Estilo y color de los bordes**, son evidentes: son las líneas que delimitan el bloque, que pueden dibujarse de varias formas en cuanto a tipo de trazo, grosor y color. Estos dos atributos también pueden actuar sobre los bordes de otros elementos contenidos dentro de un bloque, como formularios, tablas, gráficos, etc.

El **Color del fondo** se controla con las mismas instrucciones que las de la página HTML: **background**. Los parámetros de color, al igual que en HTML, se pueden escribir con su nombre (en inglés) o con la notación RGB en hexadecimal.

El **Tamaño** son las dimensiones del bloque. Si no se indican dimensiones, por defecto, el bloque ocupará todo el ancho de la ventana, y de alto lo que su contenido precise. Se utilizan dos atributos para controlarlo: **width** para el ancho y **height** para el alto.

Estilo del texto

Veamos los parámetros disponibles para dar estilo al texto. Como ya sabes, las unidades de medida aplicables a todos ellos son varias, pero para mayor claridad, en todos los ejemplos utilizaremos solamente el píxel: **px**. Sea por ejemplo la siguiente página:

```
<HTML>
<HEAD><TITLE>Estilos</TITLE>
<LINK REL="stylesheet" TYPE="text/css" HREF="estilo.css">
</HEAD>
<BODY>

Texto fuera de párrafo.<BR>
Segunda línea fuera de párrafo.

<P>texto de párrafo. <BR> segunda línea de párrafo.</P>

</BODY>
</HTML>
```

Y si escribimos la siguiente hoja de estilo **estilo.css**:

```
BODY {color:green }

P {color:red;
   font-size:20px;
   font-family:Courier;
   font-weight:bold;
   font-style:italic;
   line-height:30px;
   letter-spacing:5px;
   text-decoration:underline;
   text-transform:capitalize;
   text-align:left;
   text-indent:30px;
  }
```

Se obtiene:

Texto fuera de párrafo.
Segunda línea fuera de párrafo.

Texto De Párrafo.
Segunda Línea De Párrafo

Fíjate en la gran diferencia de estilo que hay entre las dos primeras líneas y las dos siguientes. Analicemos cómo funciona: Las dos primeras líneas de texto, en color verde, tienen todos sus valores por defecto, excepto el color que lo reciben de la etiqueta **BODY** definida en la css. Las dos siguientes, en rojo, reciben todas sus características de la etiqueta **P**, también definida en la css. Como puedes ver, hay una gran cantidad de atributos que actúan sobre el estilo de ese texto.

color:red; Este ya lo conocemos. Define el color del texto. El color deseado puede escribirse directamente, en inglés, o puede utilizarse el sistema RGB en hexadecimal. Por ejemplo, este mismo color rojo en RGB sería: #FF0000; Ciertos valores, como el rojo, pueden escribirse de forma abreviada: #F00; Otra forma de declarar el color es mediante la función rgb() que puede parametrizarse en decimal o en tantos por ciento. He aquí un ejemplo con las cuatro formas del color rojo:

```
color: #ff0000
color: #f00
color: rgb(255,0,0)
color: rgb(100%, 0%, 0%)
```

Recuerda que en el índice tienes unas tablas con los nombres y códigos de los colores.

font-size:20px; También muy conocido: define el tamaño de los caracteres. Además del tamaño definido por el usuario, expresado en cualquiera de las unidades de medida conocidas, puede tener alguno de los siguientes valores absolutos: **xx-large x-large large medium small x-small xx-small**.

font-family: Courier; Indica el nombre (o nombres) del tipo de letra que se va a emplear. En el ejemplo se ha utilizado el tipo "Courier", pero pueden escribirse varios separados por comas, lo que indica al navegador que si no existe en la máquina el primer tipo, utilice el segundo, y si tampoco, el tercero, etc. Por ejemplo: **font-family:**Courier,Verdana,Arial;

Cuando definas tipos de letra, recuerda que muchos programas y algunos drivers de impresora instalan sus propias fuentes en el sistema sin avisarte, y puede que en tu máquina haya tipos que no son estándar en Windows. No se deben utilizar tipos extraños que difícilmente se encontrarán en las máquinas de los clientes, es decir, el tipo de fuente indicado en la hoja simplemente le indica al navegador qué fuente debe usar, NO se la sirve. En la versión 3 de CSS parece que se podrá indicar al cliente dónde obtener una fuente que no tenga instalada.

font-weight:bold; Aquí se especifica el peso o grosor de la fuente. Pueden emplearse literales como **lighter**, **normal** (por defecto) o **bold**. También se pueden utilizar números entre **100** y **900**, escritos de 100 en 100 (no sirve 190, pero sí 200). Los pesos no tienen los mismos resultados en todas las máquinas, ya que depende mucho de la calidad de su pantalla, de su resolución, del navegador que emplea... En cualquier caso, no deben hacerse combinaciones extrañas, como definir un size muy pequeño y darle peso 900, ya que lo único que conseguirás es un borrón perfectamente ilegible.

font-style:italic; Solamente tiene dos posibilidades: **normal** (por defecto) o **italic** que es el empleado en el ejemplo que estamos analizando, y que hace que el texto tenga cierta inclinación.

line-height:30px; Sirve para establecer la distancia entre líneas consecutivas de un mismo párrafo. Además de en píxels o alguna de las unidades de medida que ya conocemos, puede indicarse con un simple número que viene a indicar, aproximadamente, cuantas anchuras de línea se tomarán como unidad de medida para separar una línea de otra. Así, por ejemplo, si pones **0** la segunda línea se suporpone a la primera.

letter-spacing:5px; Establece la separación entre los caracteres de la línea.

word-spacing:5px; Establece la separación entre las palabras de la línea.

text-decoration:underline; Con este parámetro, que no influye en el tamaño, puedes acompañar al texto de una delgada línea decorativa que puede estar en tres posiciones distintas, como **underline** (el típico subrayado debajo de la línea), **through** (en el centro de la línea -tachado-) o **overline** (encima del texto). Para que no aparezca se utiliza **none** (por defecto). El valor none se puede utilizar para eliminar el efecto de subrayado que ponen otros elementos, como ocurre en los links.

text-transform:capitalize; Curioso efecto que provee de cuatro posibilidades: **capitalize** que convierte en mayúscula la primera letra de cada palabra del texto, como puedes ver en el ejemplo, donde el texto original no es así. **uppercase** para convertir todas las letras a mayúsculas y **lowercase** para lo contrario, es decir, convertir todas las letras a minúsculas. El valor por defecto es **none**, que como ya habrás supuesto, deja el texto tal como está escrito.

text-align:left; Alinea el texto según convenga. Con **left** a la izquierda (por defecto), **right** a la derecha, **center** centrado, y **justify** justificado, es decir, igualando todas las líneas en longitud a izquierda o derecha.

text-indent:30px Produce que la primera línea del párrafo se escriba adentrada (indentada) un cierto espacio hacia la derecha o hacia la izquierda, dependiendo de la alineación activa.

Este sería un resumen sobre la declaración **font**:

Declaración	font-size	font-family	font-weight	font-style
Valor por defecto	medium	Definido por el usuario	normal	normal
Valores posibles	Absoluto, relativo, porcentaje, unidad de tamaño	Familia de fuente Windows	normal, lighter, bolder, un valor entre 100 y 900	normal, bold, italic
Valor porcentual	Se calcula respecto al padre	No aplicable	No aplicable	No aplicable
Se aplica a	Todos	Todos	Todos	Todos
Se hereda?	Sí	Sí	Sí	Sí

Y este sobre el resto:

Declaración	line-height	text-decoration	text-transform	text-align	text-indent	word-spacing	letter-spacing	white-space
Valor por defecto	normal	none	none	left	0	normal	normal	normal
Valores posibles	número, longitud, porcentaje, normal	none, underline, line-through, blink	capitalize, uppercase, lowercase, none, inherit	left, right, center, justify, none	longitud, porcentaje	normal, inherit, medida	normal, inherit, medida	normal, pre, nowrap, pre-wrap, pre-line, inherit
Valor porcentual	Se calcula respecto al tamaño de la fuente	No aplicable	No aplicable	No aplicable	Se calcula respecto al ancho del elemento padre	No aplicable	No aplicable	No aplicable
Se aplica a	Elementos a nivel de bloque	Todos	Todos	Elementos a nivel de bloque	Elementos a nivel de bloque	Todos	Todos	Todos
Se hereda?	Sí	No	Sí	Sí	Sí	Sí	Sí	Sí

Tenemos también otras instrucciones que pueden modificar el aspecto de determinadas zonas en lugar de actuar sobre todo el bloque.
Son los pseudo-elementos. Por ejemplo:

```
p :first-letter{
  color:red;
}
p :first-line{
  color:green;
}
p :after{
  content:"Hola";
}
p :before{
```

```
  content:"Por ejemplo:";
}
```

Como puedes ver, es fácilmente deducible lo que hace cada uno:
:first-letter actuará sobre la primera letra del párrafo.
:first-line lo hará sobre la primera línea.
:after content:"Hola"; actuará sobre el texto que haya después de la palabra "Hola" y
:before content:"Por ejemplo:"; antes de la frase "Por ejemplo:"

Bordes de un bloque

Con css se pueden definir los bordes de un bloque, que por defecto son invisibles. Por tanto, lo primero que hay que definir es el estilo del borde. Además del estilo, se puede definir su color y grosor, y todo ello puede hacerse globalmente, sobre los cuatro lados del bloque, o cada uno por separado. Las unidades de medida y los nombres o códigos de colores que pueden utilizarse, son los mismos que ya se han visto anteriormente para los textos. Para definir el grosor de los bordes tenemos:

- **border-left-width:**unidad borde izquierdo
- **border-right-width:**unidad borde derecho
- **border-top-width:**unidad borde superior
- **border-bottom-width:**unidad borde inferior
- **border-width:**unidad los cuatros bordes

Donde unidad es una de las ya conocidas, por ejemplo 1px. También pueden utilizarse constantes tales como **thin** para fino (2px), **medium** para medio (4px), y **thick** para grueso (6px).

Ningun estilo de bordes (excepto **solid**) se lleva bien con el parámetro de grosor, que implícitamente establece su dimensionado más conveniente. Tenemos:

- **border-left-style:**estilo borde izquierdo
- **border-right-style:**estilo borde derecho
- **border-top-style:**estilo borde superior
- **border-bottom-style:**estilo borde inferior
- **border-style:**estilo los cuatros bordes

Donde estilo puede ser:

- **none** (por defecto)
- **solid**
- **double**
- **ridge**
- **groove**
- **inset**
- **outset**
- **dotted**
- **dashed**

Por último, los parámetros de color de los bordes:

- **border-left-color:**color borde izquierdo
- **border-right-color:**color borde derecho
- **border-top-color:**color borde superior
- **border-bottom-color:**color borde inferior
- **border-color:**color los cuatros bordes

Donde color puede ser un nombre de color, en inglés, o su código RGB en hexadecimal, como ya sabes.

Las combinaciones de todos estos parámetros son infinitas, y lo mejor es ir haciendo pruebas. Recuerda que cada nevegador interpreta todo esto a su manera, y algunos no lo interpretan en absoluto. He aquí aplicado al ejemplo anterior sobre texto:

Si escribimos la siguiente hoja de estilo **estilo.css**:

```
BODY {color:green }

P {color:red;
   font-size:20px;
   font-family:Courier;
   font-weight:bold;
   font-style:italic;
   line-height:30px;
   letter-spacing:5px;
   text-decoration:underline;
   text-transform:capitalize;
   text-align:left;
   text-indent:30px;

   border-width:2px;
   border-color:blue;
   border-style:solid;
  }
```

Se obtiene:

Texto fuera de párrafo.
Segunda línea fuera de párrafo.

Texto De Párrafo.
Segunda Línea De Párrafo

Y este sería un resumen de su utilización:

Declaración	border-left-width border-right-width border-top-width border-bottom-width border-width	border-style	border-color border-left-color border-right-color border-top-color border-bottom-color
Valor por defecto	0	none	none
Valores posibles	Unidad de longitud, thin, medium, thick	none, solid, double, inset, outset, groove, ridge	none, color
Valor porcentual	No aplicable	No aplicable	No aplicable
Se aplica a	Todos	Todos	Todos
Se hereda?	No	No	No

Márgenes de los bloques

Los bloques, por defecto, se escriben en la ventana activa comenzando en el ángulo superior izquierdo de la misma, y su contenido, en el interior del bloque, comienza en el mismo sitio. Esto es posible modificarlo:

Para los márgenes del bloque respecto a la ventana activa:

- **margin-left:**unidad margen izquierdo
- **margin-right:**unidad margen derecho
- **margin-top:**unidad margen superior
- **margin-bottom:**unidad margen inferior
- **margin:**unidad los cuatros márgenes

Donde unidad es una de las ya conocidas.

Para los márgenes **dentro** del bloque, es decir, respecto al borde del bloque:

- **padding-left:**unidad margen interno izquierdo
- **padding-right:**unidad margen interno derecho
- **padding-top:**unidad margen interno superior
- **padding-bottom:**unidad margen interno inferior
- **padding:**unidad los cuatros márgenes internos

En resumen:

Declaración	margin margin-left margin-right margin-top margin-bottom	padding padding-left padding-right padding-top padding-bottom
Valor por defecto	0	0
Valores posibles	Longitud, porcentaje, auto	Longitud, porcentaje
Valor porcentual	Se calcula respecto al ancho del padre	Se calcula respecto al ancho del padre
Se aplica a	Todos	Todos
Se hereda?	No	No

Existe otra propiedad que consigue efectos muy parecidos a los vistos en los bordes: **outline** que sirve para crear contornos alrededor de los objetos, tales como botones, campos activos de los formularios, etc., a fin de resaltarlos, pero no se trata de bordes, sino de contornos, es decir, están ligeramente más hacia adentro, y por tanto no ocupan lugar extra en la página. La mala noticia es que no funciona en todos los navegadores.

Dimensiones de los bloques

Como ya se ha dicho más arriba, por defecto, y en función de su contenido, el bloque ocupará todo el ancho de la ventana y el alto que precise. Al componer la página puede que esto no interese y se quiera limitar el espacio ocupado por el bloque. Para ello disponemos de dos declaraciones básicas para indicar el ancho y el alto exactos que se desea, así como otras para definir el mínimo y máximo tamaño que deben alcanzar:

Declaración	width	min-width	max-width	height	min-height	max-height
Valor por defecto	auto	Incierto	none	auto	0	none
Valores posibles	Longitud, porcentaje, auto, inherit	Longitud, porcentaje, inherit	Longitud, porcentaje, none, inherit	Longitud, porcentaje, auto, inherit	Longitud, porcentaje, inherit	Longitud, porcentaje, none, inherit
Valor porcentual	Se calcula respecto al ancho del padre	Se calcula respecto al ancho del padre	Se calcula respecto al ancho del padre	Se calcula respecto al alto del padre o auto	Se calcula respecto al alto del padre	Se calcula respecto al alto del padre
Se aplica a	Todos	Todos	Todos	Todos	Todos	Todos

Se hereda?	No	No	No	No	No	No

Desafortunadamente, cada navegador interpreta las cosas de una forma, y es necesario probar con cuantos se posible. Sea, por ejemplo, la siguiente hoja de estilo **estilo.css**:

```
P {color:red;
   font-size:20px;
   font-family:Courier;
   border:2px;
   border-color:blue;
   border-style:solid;

   width:100px;
   height:100px;
  }
```

Y este texto:

<P>Esto es un ejemplo de bloque de reducidas dimensiones que ocupa solamente 100 x 100 pixels</P>

Dependiendo del navegador utilizado, esta sintaxis provocará varios resultados:

Internet Explorer

Esto es un ejemplo de bloque de reducidas dimensiones que ocupa solamente 100 x 100 pixels

Opera

Esto es un ejemplo de bloque de reducidas dimensiones que ocupa solamente 100 x 100 pixels

Netscape/Mocilla

Esto es un ejemplo de bloque de reducidas dimensiones que ocupa solamente 100 x

Como puedes ver, los tres navegadores dan preferencia al texto, y luego... hacen lo que pueden. Es evidente que la instrucción que se les ha dado no es muy coherente, y para intentar solucionar el fiasco, se puede recurrir a un nuevo atributo: **overflow** que tiene cuatro parámetros:

- **visible** Indica que el contenido no será recortado a los límites del bloque, es decir, que puede ser presentado fuera de la caja de bloque ignorando sus límites. Este valor provoca el mismo comportamiento del navegador que si no se utiliza el atributo **overflow**
- **hidden** Lo contrario del anterior. El contenido que no quepa dentro de los límites del bloque será recortado y no será visible ni accesible.
- **scroll** Mantiene las dimensiones definidas para el bloque, y provee de barras de escroll horizontal y vertical para todo el contenido que exceda los límites del bloque.
- **auto** Parecido al anterior. En lugar de crear barras de scroll tanto vertical como horizontal, incluso aunque alguna de ellas no sea necesaria, decide automáticamente cuales son precisas.

Veamos un ejemplo utilizando **scroll**:

```
P {color:red;
   font-size:20px;
```

```
    font-family:Courier;
    border:2px;
    border-color:blue;
    border-style:solid;

    width:100px;
    height:100px;

    overflow: scroll
   }
```

Se obtiene:

Si se utiliza el parámetro **hidden** puede ocurrir que interese recortar una parte del bloque en lugar de utilizar toda el área resultante. No necesariamente el área a recortar debe provenir de **hidden**, puede ser un bloque cualquiera, con **overflow** o no. Para hacer un recorte de un bloque se utiliza la propiedad **clip:** Por ejemplo:

```
clip: rect(0,100px,100px,0);
```

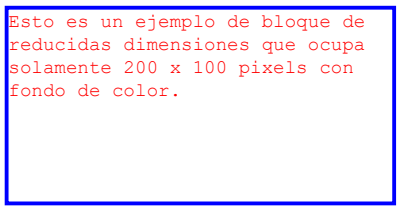
El parámetro **rect** indica la forma geométrica que se va a recortar. Actualmente sólo está disponible el **rect**ángulo. El resto del parámetro son dos ejes de coordenadas: 0,100px,100px,0 que corresponden con los vértices **superior derecho** e **inferior izquierdo** respectivamente.

Color o gráficos de fondo del bloque

Veamos por último cómo cambiar el color de fondo de los bloques. Sea, por ejemplo: **estilo.css**:

```
P {color:red;
    font-size:10px;
    font-family:Courier;
    border:2px;
    border-color:blue;
    border-style:solid;
    width:200px;
    height:100px;

    background: yellow;
   }
```



Como es habitual en HTML, los colores pueden definirse con su nombre en inglés o con la notación RGB en hexadecimal. En este caso, el color amarillo del ejemplo puede obtenerse también con **background: #FFFF00;** el color del fondo de cualquier elemento también puede ser **transparent** (por defecto). La propiedad color no se hereda, pero puede forzarse de la forma habitual: **inherit**

Si en lugar de un color le quieres poner un gráfico la sintaxis será:

background-image: url(../imgenes/fondo.gif);

Esta opción tiene tres parámetros:

- **repeat no-repeat repeat-y repeat-x** **repeat** (por defecto) hace que la imagen se repita tanto en horizontal como en vertical por todo el bloque. Dependiendo del tamaño del gráfico se puede conseguir un efecto mosaico o una sola imagen estática. Para que solamente se repita en horizontal se utiliza **repeat-x**, y para que lo haga en vertical **repeat-y**.
- **attachment: fixed attachment: scroll** , el valor **fixed** sirve para que la imagen se quede fija al hacer scroll con el texto, con lo que parece que el texto "flota" sobre la imagen. Para que la imagen acompañe al texto en su desplazamiento se utilizará **scroll** (por defecto).
- **position** Indica la posición del gráfico dentro del bloque. Si no se indica otra cosa, por defecto éste será colocado en el ángulo superior izquierdo del bloque. Para variar su posición se pueden escribir las coordenadas deseadas de la esquina superior izquierda del gráfico en píxels o en tantos por ciento respecto al total del bloque.

Estos tres parámetros adicionales (si se utilizan) hay que escribirlos **después** de haber indicado la imagen a emplear como fondo. Por ejemplo:

```
P {color:red;
   font-size:10px;
   font-family:Courier;
   border:2px;
   border-color:blue;
   border-style:solid;
   width:200px;
   height:100px;

   background-image: url(fondo.jpg);
   background-attachment: fixed;
   background-repeat: repeat;
  }
```

Esto es un ejemplo de bloque de reducidas dimensiones que ocupa solamente 200 x 100 pixels con fondo gráfico.

Gráficos y bloques flotantes

Dentro de un bloque, además de texto, pueden ir gráficos u otros bloques. Para conseguir que estos objetos se alineen respecto al texto que ya exista en el bloque, o para que se alineen varios bloques entre sí, se utiliza la instrucción **float** que puede tener tres parámetros: **left right none** (por defecto). Por ejemplo:

```
p {color:red;
   font-size:10px;
   font-family:Verdana;
   border:2px;
   border-color:blue;
   border-style:solid;
   width:200px;
   height:100px;
  }

img {float: right;}
```

Se obtiene:

texto texto texto texto texto
texto texto texto texto texto
texto texto texto texto texto
texto texto texto texto texto texto
texto texto texto texto texto texto
texto texto texto texto texto

Y por supuesto, además de posicionarlos en el bloque, los gráficos se pueden redimensionar y darles los atributos que sean necesarios para la mejor composición. Por ejemplo:

```
img {float: right;
     width:60px;
     height:60px;
     padding: 2px;}
```

texto texto texto texto
texto texto texto texto
texto texto texto texto
texto texto texto texto
texto texto texto texto
texto texto texto texto
texto texto texto texto texto texto
texto texto texto texto texto texto

Después de haber utilizado una instrucción **float** en un bloque, para cerrar su efecto sobre la alineación de los objetos hay que escribir otra en el siguiente bloque: **clear** que también tiene un parámetro con tres posibles valores: **left** **right** **both** Evidentemente, habrá que utilizar left o right según se haya definido el float anterior, y si se han incluido los dos, se utilizará **both** (ambos).

Recuerda que las instrucciones de posicionamiento y dimensionado cada navegador las interpreta a su manera, y como siempre, se deberán probar con los navegadores más habituales hasta dar con los parámetros que mejor funcionen en todos ellos (y esto puede llevar su tiempo...).

Posición y visibilidad de los bloques

Cuando un navegador carga una página, generalmente, interpretará el código HTML siguiendo el mismo orden en que fue escrito. A este orden natural de aparición de los elementos se le llama **flujo**. Viene a ser como ir apilando un elemento encima de otro al tiempo que se asigna un número de índice a cada uno, de forma que el que tenga el número más alto aparecerá más cerca del espectador. Es posible alterar el flujo de una página, es decir, que un elemento aparezca delante de otro que fue escrito **después**. Para ello simplemente hay que cambiarle el número que tiene en la pila con la propiedad **z-index = n**, donde **n** es un número entero positivo o negativo.

Cuando se altera el orden de la pila, los elementos hijos del elemento reasignado heredarán el mismo nuevo número de índice de su padre en la pila. Por ejemplo:

Sea una página que contiene dos elementos **A** y **B**. A con z-index = 4 y B con z-index = 3, A será procesado por encima de B. Supongamos que A tiene un descendiente **C** con z-index = 1. El elemento C también será procesado por encima de B aunque B tenga z-index = 3. Todos los elementos descendientes de A se situarán por encima de B y por encima de todos los descendientes de B.

La propiedad z-index solamente funciona si los elementos han sido previamente colocados utilizando una instrucción de tipo de posicionamiento: **position: valor;**

Declaración	position
Valor por defecto	static
Valores posibles	static relative

	absolute fixed inherit
Valor porcentual	No aplicable
Se aplica a	Todos
Se hereda?:	No

Si se utiliza **static**, como su nombre indica, no se puede modificar la posición natural de los elementos. Si se quiere mover un elemento de su posición natural hay que utilizar los valores **relative** o **absolute**.

En este ejemplo de posicionamiento relativo, al segundo elemento se le aplica un desplazamiento de 5 px por arriba y por la izquierda, lo que da por resultado que invada parte del espacio del elemento 3 y se salga del contenedor por la derecha, es decir, que los posicionamientos relativos de un elemento solamente afectan a ese elemento, mientras que el resto continua manteniendo sus posiciones normales en el flujo.

BLOQUE 1

BLOQUE 2

BLOQUE 3

Este es el código necesario:

```
<div style="position: relative; border: 2px solid red; width: 200px;">
   <div style="background:  #0000ff; color: white;"> BLOQUE 1</div>
   <div style="position: relative; top: 5px; left: 5px;
         background: #00a0a0; color: white;">BLOQUE 2</div>
   <div style="background: #ffc0c0; color: white;">BLOQUE 3</div>
</div>
```

Y puede que te preguntes de dónde han salido las instrucciones **top** y **left**. Pues son dos de las cuatro direcciones en que se pueden mover los bloques, y junto con **bottom** y **rigth** nos permitirán indicar la dirección de los movimientos tanto absolutos como relativos. Fíjate en que estas cuatro instrucciones no indican en qué dirección se moverá el elemento, sino el espacio que se dejará libre antes de procesar el elemento.

Veamos ahora qué ocurre si el desplazamiento indicado es **absolute**:

BLOQUE 1
BLOQUE 3 BLOQUE 2

Como puedes ver, en este caso el espacio abandonado por el elemento 2 no queda libre, y ha sido optimizado por el bloque contenedor, superponiéndose el bloque 2 sobre el 1 y el 3. El bloque 2, además, ha perdido la longitud que heredaba del bloque contenedor, y que si se quiere mantener, será necesario forzar la herencia con **width: inherit;**

Este es el código:

```
<div style="position: relative; border: 2px solid red; width: 200px;">
   <div style="background:  #0000ff; color: white;"> BLOQUE 1</div>
   <div style="position: absolute; top: 10px; left: 85px; width: inherit;
         background: #00a0a0; color: white;">BLOQUE 2 </div>
   <div style="background: #ffc0c0; color: white;">BLOQUE 3</div>
</div>
```

Se dice que un bloque está posicionado si su propiedad **position** es distinta de **static**, como en los dos ejemplos anteriores. Las direcciones de desplazamiento pueden ser: **top, right, bottom, left**

Declaraciones	top right

	bottom left
Valor por defecto	auto
Valores posibles	auto inherit valor porcentual medida
Valor porcentual	Referido a la altura del bloque contenedor
Se aplica a	Bloques posicionados
Se hereda?:	No

Veamos ahora otra propiedad muy interesante de los bloques que permite que sean visibles o no: se trata de **display: valor;** donde **valor** puede ser, entre otros:

Declaración	display
Valor por defecto	inline
Valores posibles	inline none inherit block list-item table inline-table table-row-group table-header-group table-footer-group table-row table-column-group table-column table-cell table-caption
Valor porcentual	No aplicable
Se aplica a	Todos
Se hereda?:	No

Cualquiera de los valores posibles (excepto **none**) generará uno o varios bloques. Al aplicar el valor **none** se consigue la desaparición del bloque, incluido su espacio en la página. Los elementos descendientes tampoco serán visibles ni ocuparán espacio, es decir, no se trata de bloques invisibles, sino inexistentes a todos los efectos. Esa es la diferencia con las propiedades sobre visibilidad, que se describen a continuación, que provocan que un elemento pueda ser invisible pero siga ocupando un espacio en la página.

La siguiente propiedad, **visibility: valor** como su nombre indica, permite controlar la visibilidad de un bloque, pero siempre conservando en la página su espacio físico inicial. En realidad se hace transparente, lo que permite utilizarlo para ocultar o destapar otros bloques.

Declaración	visibility
Valor por defecto	visible
Valores posibles	visible hidden

	inherit collapse
Valor porcentual	No aplicable
Se aplica a	Bloques
Se hereda?:	No

CSS + JavaScript

La combinación de CSS y JavaScript es una poderosa herramienta para que las páginas resulten mucho más dinámicas.

En este ejemplo se consigue que un bloque sea sustituido por otro al pulsar el botón de un formulario. Pude hacerse con un link, o con cualquier otro elemento. Solamente hay que capturar el evento correspondiente.

BLOQUE 1
BLOQUE 2
BLOQUE 3

Ver código | Ver bloque

En este otro ejemplo se elimina un bloque de la página al pulsar sobre un párrafo, aunque podría ser cualquier otro elemento. Aunque lo parezca, no es igual que el anterior, ya que aquí el espacio del bloque desaparecido es recuperado.

Pulsar aquí para ocultar o mostrar el párrafo siguiente.

Este bloque aparece y desaparece de la página.

Este es un ejemplo de la capacidad de CSS + JavaScript para ocultar bloques.

Veamos el código JavaScript necesario:

```
function Alternar(Seccion){
    if (Seccion.style.display=="none"){Seccion.style.display=""}
    else{Seccion.style.display="none"}
}
```

Aquí está el código HTML:

```
<p style="cursor:s-resize" onClick="Alternar(seccion1)">
   Pulsar aquí para ocultar o mostrar el párrafo siguiente.
</p>

<div id="seccion1" style="background: #ffc0c0; color: white; display: block">
   Este bloque aparece y desaparece de la página.
</div>
```

Y aquí la hoja de estilo.

```
ParentDiv01 .ChildDivOrig {display: block;}
.ParentDiv01 .ChildDivNew {display: none;}
.ParentDiv02 .ChildDivOrig {display: none;}
.ParentDiv02 .ChildDivNew {display: block;}
```

Estilo de las listas

Es posible definir el tipo de las listas con CSS. Por ejemplo:

```
ul {list-style-type:square;}
ol {list-style-type:upper-roman;}
```

El parámetro **list-style-type** puede tener, entre otros, los siguientes valores:

- **disc** (por defecto)
- **none**
- **circle**
- **square**
- **decimal**
- **upper-roman**
- **lower-roman**
- **upper-alpha**
- **lower-alpha**

Como puedes ver, aunque denominados de otra forma, coinciden con los definidos en la sección de las listas vistas en HTML.

Estilos en los links

Hay un elemento HTML que genera un estilo propio por defecto: **<A>**. Como ya sabes, los enlaces aparecen por defecto de otro color y subrayados, y cambian de color según hayan sido visitados o no. Estos efectos de estilo no definidos por el usuario se llaman **pseudoclases**. Hay dos posibles modificaciones de estilo para este elemento; el color del enlace en sus distintos estados y si aparece subrayado o no. Teóricamente no debiera influir el orden en que se escriban los distintos estados en la CSS, pero es mejor escribir las declaraciones en el siguiente orden (por supuesto, los colores son libres):

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

a:link {color: #FF0000;}
a:visited {color: #00FF00;}
a:hover {color: #FF00FF;}
a:active {color: #0000FF;}
```

Donde **link** indica que el enlace no ha sido visitado, **visited** que el enlace ha sido visitado, **active** que el enlace está siendo pulsado, y **hover** que el ratón está pasando sobre él (sin pulsar).

Estilos en los formularios

Todo lo visto sobre los estilos, también es aplicable a los elementos de un formulario, bien desde las hojas CSS o en el código HTML del propio formulario. Por ejemplo el formulario:

Borde simple | Borde doble | Solo lectura sin borde | Botón transparente

Se escribe así:

```
<FORM METHOD="GET" ACTION="">

   <INPUT TYPE="text" STYLE="background:yellow;color:red" NAME="campo1"
       VALUE="Borde simple">

   <INPUT TYPE="text" STYLE="border:double;background:yellow;color:blue" NAME="campo2"
       VALUE="Borde doble">

   <INPUT TYPE="text" STYLE="border:0;background:yellow;color:blue" readonly NAME="campo3"
       VALUE="Solo lectura sin borde">

   <INPUT TYPE="button" STYLE="background:transparent;color:red"
       VALUE="Botón transparente">
```

```
</FORM>
```

Este formulario tiene las instrucciones de estilo embebidas en el código html. Su equivalente utilizando hoja css sería:

```
<FORM METHOD="GET" ACTION="">

   <INPUT CLASS="controles1"  TYPE="text" NAME="campo1"  VALUE="Borde simple" >

   <INPUT CLASS="controles2"  TYPE="text" NAME="campo2"VALUE="Borde doble">

   <INPUT CLASS="controles3" TYPE="text" readonly NAME="campo3" VALUE="Solo lectura sin borde">

   <INPUT CLASS="botones"  TYPE="button" VALUE="Botón transparente">

</FORM>
```

Y esta sería la hoja de estilo correspondiente:

```
/* Hoja de estilo del formulario */
.controles1 {background:yellow;color:red;}
.controles2 {border:double;background:yellow;color:blue}
.controles3 {border:0;background:yellow;color:blue}
.botones {background:transparent;color:red}
```

En los formularios se puede cambiar el estilo de los bordes de los controles, pero no el grosor como se hace en los bloques de texto. Un formulario se puede considerar un bloque, por lo que también es posible cambiar el color del fondo, definir bordes, etc. Por supuesto, todo esto también es aplicable a las tablas, imágenes y al resto de elementos html.

Efectos especiales con el texto

Algunas veces habrás visto páginas con textos que parecen gráficos. Algunos tipos, en efecto, solamente se pueden conseguir con editores gráficos, pero algunas veces son simples efectos de estilo. Veamos un ejemplo:

```
all.texto {
  margin-top:-24px;
  color:yellow;
  font-size:20px;
  font-family:Verdana;
}
all.sombra {
  margin-top:2px;
  margin-left:2px;
  color:black;
  font-size:20px;
  font-family:Verdana;
}

<DIV ALIGN="CENTER" CLASS="sombra">Prueba de texto sombreado</DIV>
<DIV ALIGN="CENTER" CLASS="texto">Prueba de texto sombreado</DIV>
```

Se obtiene:

Prueba de texto sombreado

Y si en lugar de escribir una hoja css prefieres hacerlo con estilo embebido en el HTML, este es el código:

```
<DIV ALIGN="CENTER" STYLE="margin-top:2px; margin-left:2px; color:black;
font-size:20px; font-family:Verdana;">Prueba de texto sombreado</DIV>

<DIV ALIGN="CENTER" STYLE="margin-top:-24px; color:yellow; font-size:20px;
font-family:Verdana;">Prueba de texto sombreado</DIV>
```

Tablas

Las tablas generadas con HTML tienen un serio problema de estética. En efecto, si los bordes son visibles, su estilo es bastante tosco y pueden arruinar cualquier diseño. Existen instrucciones en CSS para conseguir que las tablas tengan bastante mejor aspecto. Se puede modificar cualquiera de sus componentes: bordes de la tabla, bordes de las celdas, fondo, título, alineación, etc.

Declaración	caption	empty-cells	border-collapse	border-spacing	table-layout	white-space
Valor por defecto	top	show	separate	0 0	auto	normal
Valores posibles	top bottom	show hide inherit	separate collapse	cualquier unidad de medida	auto fixed	normal pre nowrap
Valor porcentual	No aplicable	No aplicable	No aplicable	No aplicable	No aplicable	No aplicable
Se aplica a	table	td th	Todos	Todos	table	Todos
Se hereda?:	No	Si	Si	Si	No	No

Además de estas declaraciones específicas están disponibles todas las que afectan a los bordes, tanto de la tabla como de las celdas. Las combinaciones posibles de todas ellas son tantas, que lo mejor es verlas sobre la marcha con el magnífico asistente diseñado por SONACON. El código obtenido, como siempre, es conveniente probarlo en los navegadores más comunes, ya que suele haber notables diferencias de interpretación entre ellos.

[Indice]
