**Propiedad clear**

|  |  |
| --- | --- |
| **Definición** | Controla el comportamiento de los elementos adyacentes a elementos posicionados de forma flotante |
| **Valores  permitidos** | Alguno o todos los siguientes valores y en cualquier orden:   * none * left * right * both * [inherit](http://librosweb.es/referencia/css/inherit.html) |
| **Valor inicial** | none |
| **Se aplica a** | Los elementos de bloque |
| **Válida en** | [medios visuales](http://librosweb.es/referencia/css/medios.html) |
| **Se hereda** | no |
| **Definición en el estándar** | [w3.org/TR/CSS2/visuren.html#propdef-clear](http://www.w3.org/TR/CSS2/visuren.html#propdef-clear) |

**Ejemplos de uso**

La propiedad clear indica el lado o lados de una caja que no debe ser adyacente a un elemento posicionado de forma flotante con la propiedad [float](http://librosweb.es/referencia/css/float.html). Los cuatro valores permitidos para esta propiedad tienen el siguiente significado:

* left, hace que la caja sobre la que se aplica baje hasta que su borde superior esté por debajo del borde inferior de cualquier elemento flotado a la izquierda.
* right, hace que la caja sobre la que se aplica baje hasta que su borde superior esté por debajo del borde inferior de cualquier elemento flotado a la derecha.
* both, hace que la caja sobre la que se aplica baje hasta que su borde superior esté por debajo del borde inferior de cualquier elemento flotado a la izquierda o a la derecha.
* none, es el valor por defecto que se aplica a todos los elementos y no tiene efecto sobre la posición de la caja.

Considera el siguiente ejemplo de página que contiene una imagen y dos párrafos de texto:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sit amet feugiat justo, vitae feugiat purus. Vivamus aliquam ligula at ornare dignissim. Ut imperdiet rutrum tincidunt. Phasellus rhoncus eros auctor porta vulputate.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis posuere nibh massa, nec sollicitudin eros vestibulum non. Aliquam pellentesque pharetra tortor ut feugiat. Donec vitae gravida nunc, eget tincidunt magna. Aliquam accumsan et justo quis porttitor.

Si se aplica la propiedad [float](http://librosweb.es/referencia/css/float.html) con el valor left a la imagen, el resultado será el siguiente:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sit amet feugiat justo, vitae feugiat purus. Vivamus aliquam ligula at ornare dignissim. Ut imperdiet rutrum tincidunt. Phasellus rhoncus eros auctor porta vulputate.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis posuere nibh massa, nec sollicitudin eros vestibulum non. Aliquam pellentesque pharetra tortor ut feugiat. Donec vitae gravida nunc, eget tincidunt magna. Aliquam accumsan et justo quis porttitor.

Si no quieres que los contenidos del primer párrafo fluyan alrededor de la imagen flotante, puedes aplicar los siguientes estilos CSS:

p {

clear: left;

}

Ahora la caja del párrafo bajará hasta que su borde superior se encuentre por debajo del borde inferior del elemento flotado, es decir, por debajo del borde inferior de la imagen:

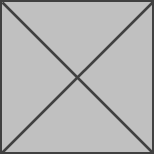


Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sit amet feugiat justo, vitae feugiat purus. Vivamus aliquam ligula at ornare dignissim. Ut imperdiet rutrum tincidunt. Phasellus rhoncus eros auctor porta vulputate.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis posuere nibh massa, nec sollicitudin eros vestibulum non. Aliquam pellentesque pharetra tortor ut feugiat. Donec vitae gravida nunc, eget tincidunt magna. Aliquam accumsan et justo quis porttitor.

Si en el ejemplo anterior se hubiera aplicado la propiedad clear: right, no se habrían producido cambios, ya que no hay ningún elemento flotado a la derecha. Asimismo, si se hubiera aplicado la propiedad clear: both, el resultado habría sido el mismo.

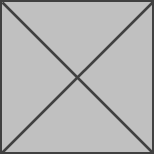
En vez de aplicar la propiedad clear: left a todos los párrafos de la página, resulta habitual dejar que las imágenes floten alrededor de algunos párrafos, pero evitar que lo hagan en el resto:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sit amet feugiat justo, vitae feugiat purus. Vivamus aliquam ligula at ornare dignissim. Ut imperdiet rutrum tincidunt. Phasellus rhoncus eros auctor porta vulputate.

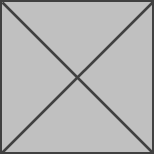
(clear: left) Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis posuere nibh massa, nec sollicitudin eros vestibulum non. Aliquam pellentesque pharetra tortor ut feugiat. Donec vitae gravida nunc, eget tincidunt magna. Aliquam accumsan et justo quis porttitor.

Si en el ejemplo anterior hubiera elementos flotados tanto a la izquierda como a la derecha, el segundo párrafo debería utilizar la propiedad clear: both en vez de clear: left:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sit amet feugiat justo, vitae feugiat purus. Vivamus aliquam ligula at ornare dignissim. Ut imperdiet rutrum tincidunt. Phasellus rhoncus eros auctor porta vulputate.

(clear: both) Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis posuere nibh massa, nec sollicitudin eros vestibulum non. Aliquam pellentesque pharetra tortor ut feugiat. Donec vitae gravida nunc, eget tincidunt magna. Aliquam accumsan et justo quis porttitor.



Etiam egestas, orci ac venenatis iaculis, urna nunc fringilla turpis, a faucibus purus metus quis libero. Maecenas malesuada eu elit eget pretium. Maecenas congue lorem nisl, vel interdum neque convallis quis. Etiam a laoreet augue.

**4.2. Propiedad display**

|  |  |
| --- | --- |
| **Definición** | Establece el tipo de caja generada por un elemento |
| **Valores  permitidos** | Uno y sólo uno de los siguientes valores:   * inline * block * list-item * run-in * inline-block * table * inline-table * table-row-group * table-header-group * table-footer-group * table-row * table-column-group * table-column * table-cell * table-caption * none * [inherit](http://librosweb.es/libro/css_avanzado/capitulo_4/inherit.html) |
| **Valor inicial** | inline |
| **Se aplica a** | Todos los elementos |
| **Válida en** | all |
| **Se hereda** | no |
| **Definición en el estándar** | [w3.org/TR/CSS21/visuren.html#propdef-display](http://www.w3.org/TR/CSS21/visuren.html#propdef-display) |

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

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

El valor más sencillo de display es none que hace que el elemento no genere ninguna caja. El resultado es que el elemento desaparece por completo de la página y no ocupa sitio, por lo que los elementos adyacentes ocupan su lugar. Si se utiliza la propiedad display: none sobre un elemento, todos sus descendientes también desaparecen por completo de la página.

Si se quiere hacer un elemento invisible, es decir, que no se vea pero que siga ocupando el mismo sitio, se debe utilizar la propiedad visibility. La propiedad display: none se utiliza habitualmente en aplicaciones web dinámicas creadas con JavaScript y que muestran/ocultan contenidos cuando el usuario realiza alguna acción como pulsar un botón o un enlace.

Los otros dos valores más utilizados son block e inline que hacen que la caja de un elemento sea de bloque o en línea respectivamente. El siguiente ejemplo muestra un párrafo y varios enlaces a los que se les ha añadido un borde para mostrar el espacio ocupado por cada caja:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

[Lorem ipsum](http://librosweb.es/libro/css_avanzado/capitulo_4/propiedad_display.html) [Donec mollis nunc in leo](http://librosweb.es/libro/css_avanzado/capitulo_4/propiedad_display.html) [Vivamus fermentum](http://librosweb.es/libro/css_avanzado/capitulo_4/propiedad_display.html)

Como el párrafo es por defecto un elemento de bloque (*"block element"*), ocupa todo el espacio disponible hasta el final de su línea, aunque sus contenidos no ocupen todo el sitio. Por su parte, los enlaces por defecto son elementos en línea (*"inline element"*), por lo que su caja sólo ocupa el espacio necesario para mostrar sus contenidos.

Si se aplica la propiedad display: inline al párrafo del ejemplo anterior, su caja se convierte en un elemento en línea y por tanto sólo ocupa el espacio necesario para mostrar sus contenidos:

**[display: inline]** Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

[Lorem ipsum](http://librosweb.es/libro/css_avanzado/capitulo_4/propiedad_display.html) [Donec mollis nunc in leo](http://librosweb.es/libro/css_avanzado/capitulo_4/propiedad_display.html) [Vivamus fermentum](http://librosweb.es/libro/css_avanzado/capitulo_4/propiedad_display.html)

Para visualizar más claramente el cambio en el tipo de caja, el siguiente ejemplo muestra un mismo párrafo largo con display: block y display: inline:

**[display: block]** Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede. Nulla cursus porta sem. Donec mollis nunc in leo.

**[display: inline]** Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede. Nulla cursus porta sem. Donec mollis nunc in leo.

De la misma forma, si en los enlaces del ejemplo anterior se emplea la propiedad display: block se transforman en elementos de bloque, por lo que siempre empiezan en una nueva línea y siempre ocupan todo el espacio disponible en la línea, aunque sus contenidos no ocupen todo el sitio:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

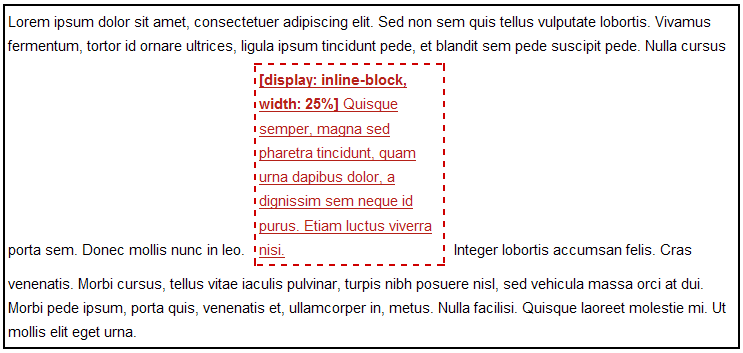
[**[display: block]** Lorem ipsum**[display: block]** Donec mollis nunc in leo**[display: block]** Vivamus fermentum](http://librosweb.es/libro/css_avanzado/capitulo_4/propiedad_display.html)

Uno de los valores más curiosos de display es inline-block, que crea cajas que son de bloque y en línea de forma simultánea. Una caja de tipo inline-block se comporta como si fuera de bloque, pero respecto a los elementos que la rodean es una caja en línea.

El enlace del siguiente ejemplo es de tipo inline-block, lo que permite por ejemplo establecerle un tamaño mediante la propiedad width:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede. Nulla cursus porta sem. Donec mollis nunc in leo. [**[display: inline-block, width: 25%]**Quisque semper, magna sed pharetra tincidunt, quam urna dapibus dolor, a dignissim sem neque id purus. Etiam luctus viverra nisi.](http://librosweb.es/libro/css_avanzado/capitulo_4/propiedad_display.html) Integer lobortis accumsan felis. Cras venenatis. Morbi cursus, tellus vitae iaculis pulvinar, turpis nibh posuere nisl, sed vehicula massa orci at dui. Morbi pede ipsum, porta quis, venenatis et, ullamcorper in, metus. Nulla facilisi. Quisque laoreet molestie mi. Ut mollis elit eget urna.

Si tu navegador soporta el valor inline-block, el ejemplo anterior se debe visualizar de forma similar a la siguiente imagen:



**Figura 4.1** Ejemplo del valor inline-block de la propiedad display

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

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Sed non sem quis tellus vulputate lobortis.

Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede.

A continuación se muestra el código HTML del ejemplo anterior:

<p style="display: list-item; margin-left: 2em">Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>

<p style="display: list-item; margin-left: 2em">Sed non sem quis tellus vulputate lobortis.</p>

<p style="display: list-item; margin-left: 2em">Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede.</p>

Los elementos con la propiedad display: list-item son exactamente iguales que los elementos <li> a efectos de su visualización, por lo que se pueden utilizar las propiedades de listas como list-style-type, list-style-image, list-style-position y list-style.

Otro de los valores curiosos de la propiedad display es run-in, que genera una caja de bloque o una caja en línea dependiendo del contexto, es decir, dependiendo de sus elementos adyacentes. El comportamiento de las cajas de tipo run-in se rige por las siguientes reglas:

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

El siguiente ejemplo muestra una misma caja de tipo run-in que se visualiza de forma muy diferente en función del tipo de caja que existe a continuación:

**[display: run-in]** Lorem ipsum

**[display: block]** dolor sit amet, consectetuer adipiscing elit.

**[display: run-in]** Lorem ipsum

**[display: inline]** dolor sit amet, consectetuer adipiscing elit.

El código HTML y CSS del ejemplo anterior se muestra a continuación:

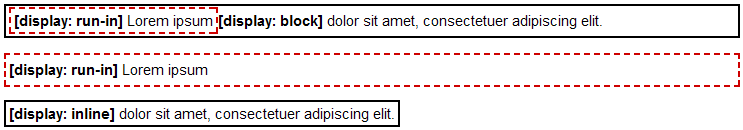
<p style="display: run-in; border: 2px dashed #C00;"><strong>[display: run-in]</strong> Lorem ipsum</p>

<p style="display: block; border: 2px solid #000;"><strong>[display: block]</strong> dolor sit amet, consectetuer adipiscing elit.</p>

<p style="display: run-in; border: 2px dashed #C00;"><strong>[display: run-in]</strong> Lorem ipsum</p>

<p style="display: inline; border: 2px solid #000;"><strong>[display: inline]</strong> dolor sit amet, consectetuer adipiscing elit.</p>

En la actualidad sólo la última versión del navegador Opera es capaz de mostrar correctamente el ejemplo anterior, tal y como muestra la siguiente imagen:



**Figura 4.2** Ejemplo del valor run-in de la propiedad display

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

<h3 style="display: run-in">Lorem ipsum dolor sit amet</h3>

<p>Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede. Nulla cursus porta sem. Donec mollis nunc in leo. Integer lobortis accumsan felis.</p>

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

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

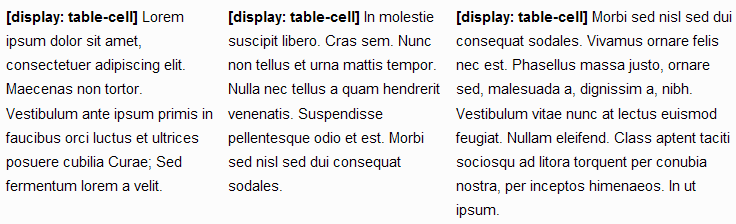
**[display: table-cell]**Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas non tortor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Sed fermentum lorem a velit.

**[display: table-cell]** In molestie suscipit libero. Cras sem. Nunc non tellus et urna mattis tempor. Nulla nec tellus a quam hendrerit venenatis. Suspendisse pellentesque odio et est. Morbi sed nisl sed dui consequat sodales.

**[display: table-cell]** Morbi sed nisl sed dui consequat sodales. Vivamus ornare felis nec est. Phasellus massa justo, ornare sed, malesuada a, dignissim a, nibh. Vestibulum vitae nunc at lectus euismod feugiat. Nullam eleifend. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. In ut ipsum.

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

Si utilizas un navegador con soporte completo de CSS 2.1, el ejemplo anterior se visualiza tal y como muestra la siguiente imagen:



**Figura 4.3** Ejemplo del valor table-cell de la propiedad display

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

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

Además, la estructura creada con una tabla nunca se rompe, ya que las celdas de datos nunca se visualizan una debajo de otra cuando la ventana del navegador se hace muy pequeña. Sin embargo, cuando se define la estructura mediante elementos <div> posicionados es posible que la página se rompa y alguna columna se muestre debajo de otros contenidos.

Utilizando la propiedad display de forma avanzada es posible crear una estructura de página que sea semánticamente correcta, esté diseñada exclusivamente con CSS y que se comporte exactamente igual que como lo hacen las tablas.

El siguiente código HTML corresponde a la estructura de una página con tres columnas:

...

<div id="contenedor">

<div id="contenidos">

<div id="secundario">

Curabitur rutrum...

</div>

<div id="principal">

Lorem ipsum dolor sit amet...

</div>

<div id="lateral">

Nam erat massa...

</div>

</div>

</div>

...

Utilizando las siguientes reglas CSS y la propiedad display es posible hacer que los elementos <div> anteriores se comporten como si fueran elementos <tr> y <td>:

#contenedor {

display: table;

border-spacing: 5px;

}

#contenidos {

display: table-row;

}

#principal, #secundario, #lateral {

display: table-cell;

}

#principal {

width: 60%;

}

#secundario, #lateral {

width: 20%;

}

El elemento #contenedor se visualiza como una tabla porque se le aplica la propiedad display: table. De esta forma, se pueden aplicar al elemento #contenedor propiedades exclusivas de las tablas como border-spacing. El elemento #contenidos se visualiza como si fuese una fila de tabla (etiqueta <tr>). En su interior se encuentran las tres columnas de la página que se visualizan como si fueran tres elementos <td> gracias a la propiedad display: table-cell.

A continuación se muestra el resultado obtenido al aplicar estas reglas CSS al código HTML anterior:

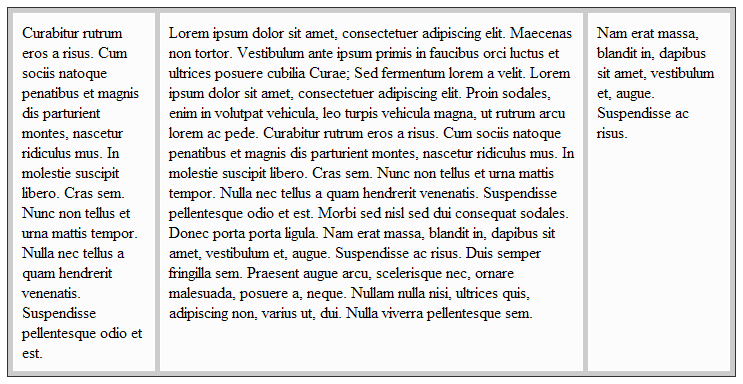
Curabitur rutrum eros a risus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. In molestie suscipit libero. Cras sem. Nunc non tellus et urna mattis tempor. Nulla nec tellus a quam hendrerit venenatis. Suspendisse pellentesque odio et est.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas non tortor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Sed fermentum lorem a velit. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin sodales, enim in volutpat vehicula, leo turpis vehicula magna, ut rutrum arcu lorem ac pede. Curabitur rutrum eros a risus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. In molestie suscipit libero. Cras sem. Nunc non tellus et urna mattis tempor. Nulla nec tellus a quam hendrerit venenatis. Suspendisse pellentesque odio et est. Morbi sed nisl sed dui consequat sodales. Donec porta porta ligula. Nam erat massa, blandit in, dapibus sit amet, vestibulum et, augue. Suspendisse ac risus. Duis semper fringilla sem. Praesent augue arcu, scelerisque nec, ornare malesuada, posuere a, neque. Nullam nulla nisi, ultrices quis, adipiscing non, varius ut, dui. Nulla viverra pellentesque sem.

Nam erat massa, blandit in, dapibus sit amet, vestibulum et, augue. Suspendisse ac risus.

La estructura de la página del ejemplo anterior está diseñada exclusivamente con CSS pero se comporta como si fuera una tabla. Todas las columnas de la página tienen la misma altura sin necesidad de recurrir a ningún truco y la página nunca se rompe por muy pequeña que se haga la ventana del navegador.

Si visualizas esta página con un navegador que soporte correctamente la propiedad display de CSS 2.1, el ejemplo anterior se ve tal y como muestra la siguiente imagen:



**Figura 4.4** Ejemplo de los valores table, table-row y table-cell de la propiedad display

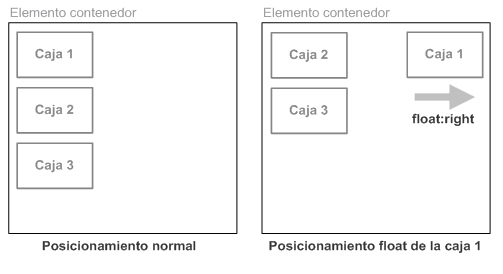
Por último, aunque el estándar CSS 2.1 establece que el valor por defecto de la propiedad display es inline, todos los navegadores obvian esta recomendación y asignan por defecto el valor block a los elementos de bloque y el valor inline a los elementos en línea.

**5.7. Posicionamiento flotante**

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

Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una *caja flotante*, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.

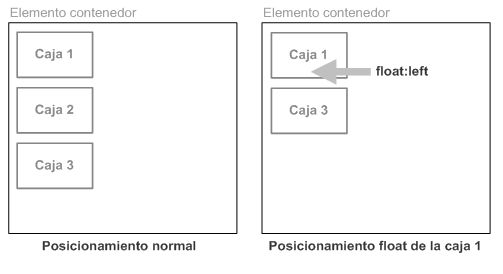
La siguiente imagen muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1:



**Figura 5.13** Ejemplo de posicionamiento float de una caja

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

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

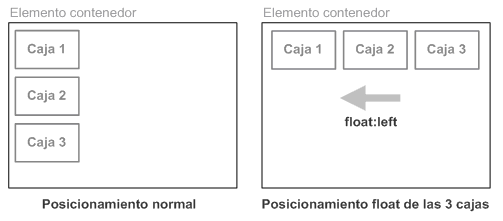


**Figura 5.14** Ejemplo de posicionamiento float de una caja

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

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

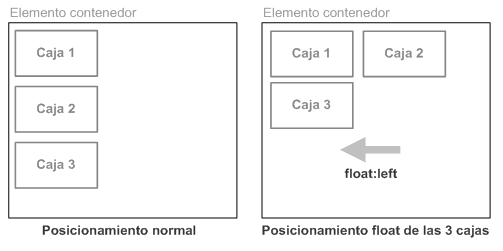
Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible. En el siguiente ejemplo se posicionan de forma flotante hacia la izquierda las tres cajas:



**Figura 5.15** Ejemplo de posicionamiento float de varias cajas

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

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



**Figura 5.16** Ejemplo de posicionamiento float cuando no existe sitio suficiente

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

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

| **Propiedad** | **float** |
| --- | --- |
| **Valores** | left | right | none | [inherit](https://librosweb.es/referencia/css/el_valor_inherit.html) |
| **Se aplica a** | Todos los elementos |
| **Valor inicial** | none |
| **Descripción** | Establece el tipo de posicionamiento flotante del elemento |

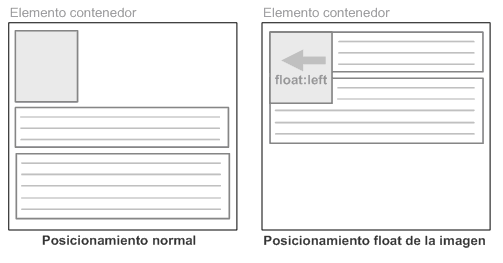
Si se indica un valor left, la caja se desplaza hasta el punto más a la izquierda posible en esa misma línea (si no existe sitio en esa línea, la caja baja una línea y se muestra lo más a la izquierda posible en esa nueva línea). El resto de elementos adyacentes se adaptan y *fluyen* alrededor de la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

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

Ejercicio 6

[Ver enunciado](http://librosweb.es/css/capitulo_15/ejercicio_6.html)

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



**Figura 5.17** Elementos que fluyen alrededor de un elemento posicionado mediante float

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

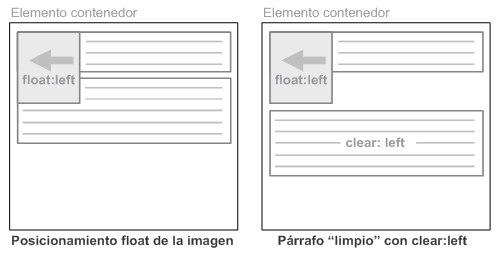
img {

float: left;

}

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

CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante float. De hecho, en muchas ocasiones es admisible que algunos contenidos fluyan alrededor de una imagen, pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen:



**Figura 5.18** Forzando a que un elemento no fluya alrededor de otro elemento posicionado mediante float

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

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

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

| **Propiedad** | **clear** |
| --- | --- |
| **Valores** | none | left | right | both | [inherit](https://librosweb.es/referencia/css/el_valor_inherit.html) |
| **Se aplica a** | Todos los elementos de bloque |
| **Valor inicial** | none |
| **Descripción** | Indica el lado del elemento que no debe ser adyacente a ninguna caja flotante |

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

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

Si se indica el valor right, el comportamiento es análogo, salvo que en este caso se tienen en cuenta los elementos desplazados hacia la derecha.

El valor both despeja los lados izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante hacia la izquierda o hacia la derecha.

Como se verá más adelante, la propiedad clear es imprescindible cuando se crean las estructuras de las páginas web complejas.

Si se considera el siguiente código CSS y HTML:

#paginacion {

border: 1px solid #CCC;

background-color: #E0E0E0;

padding: .5em;

}

.derecha { float: right; }

.izquierda { float: left; }

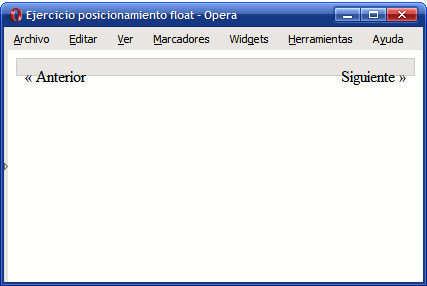
<div id="paginacion">

<span class="izquierda">&laquo; Anterior</span>

<span class="derecha">Siguiente &raquo;</span>

</div>

Si se visualiza la página anterior en cualquier navegador, el resultado es el que muestra la siguiente imagen:



**Figura 5.19** Visualización incorrecta de dos elementos posicionados mediante float

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

La solución consiste en utilizar la propiedad overflow (que se explica más adelante) sobre el elemento contenedor:

#paginacion {

border: 1px solid #CCC;

background-color: #E0E0E0;

padding: .5em;

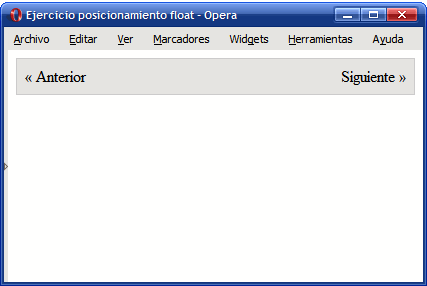
overflow: hidden;

}

.derecha { float: right; }

.izquierda { float: left; }

Si se visualiza de nuevo la página anterior en cualquier navegador, el resultado ahora sí que es el esperado:



**Figura 5.20** Visualización correcta de dos elementos posicionados mediante float

**Propiedad height**

|  |  |
| --- | --- |
| **Definición** | Establece la altura de los elementos de bloque |
| **Valores  permitidos** | Uno y sólo uno de los siguientes valores:   * [medida](http://librosweb.es/referencia/css/unidades_de_medida.html) * [porcentaje](http://librosweb.es/referencia/css/porcentajes.html) * auto * [inherit](http://librosweb.es/referencia/css/inherit.html) |
| Si se utilizan porcentajes, hace referencia a la altura del elemento en el que se encuentra. Si este elemento contenedor no tiene establecida una altura de forma explícita y el elemento para el que se define la altura no está posicionado de forma absoluta, se ignora la altura en porcentaje y se sustituye por el valor auto |
| **Valor inicial** | auto |
| **Se aplica a** | Todos los elementos, salvo los elementos en línea que no sean imágenes, las columnas de tabla y los grupos de columnas de tabla. |
| **Válida en** | [medios visuales](http://librosweb.es/referencia/css/medios.html) |
| **Se hereda** | no |
| **Definición en el estándar** | [w3.org/TR/CSS21/visudet.html#propdef-height](http://www.w3.org/TR/CSS21/visudet.html#propdef-height) |

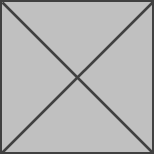
**Ejemplos de uso**

Para los elementos de bloque y las imágenes, la propiedad height permite establecer su altura directamente mediante una [unidad de medida](http://librosweb.es/referencia/css/medidas). El siguiente ejemplo establece la altura a un <div> para que no se muestre con la altura por defecto calculada por el navegador:

div { height: 100px; }

Las imágenes son el único elemento en línea para el que se tiene en cuenta la propiedad height. Por lo tanto, establecer la propiedad height de CSS es equivalente a establecer el atributo height de HTML.

El siguiente ejemplo muestra la imagen con una altura de 80px, independientemente de la altura real que tenga la imagen:



La propiedad height no funciona con los elementos en línea que no son imágenes. En el siguiente ejemplo, a pesar de que a cada enlace se le asigna una altura diferente, el navegador los muestra todos con la misma altura:

[(height: 20px) Lorem ipsum dolor sit amet](http://librosweb.es/referencia/css/height.html)   
  
[(height: 40px) Lorem ipsum dolor sit amet](http://librosweb.es/referencia/css/height.html)   
  
[(height: 60px) Lorem ipsum dolor sit amet](http://librosweb.es/referencia/css/height.html)

Si se utilizan unidades de medida, los valores indicados no pueden ser negativos. El uso de [porcentajes](http://librosweb.es/referencia/css/porcentajes.html) para definir la altura de los elementos no es tan sencillo como cuando se utiliza para definir su anchura.

Cuando un elemento establece su altura mediante un porcentaje, la referencia que se toma es la altura de su elemento contenedor. Sin embargo, si ese elemento contenedor no tiene establecida una altura de forma explícita, se ignora la altura en porcentaje y se sustituye por el valor auto.

En el siguiente ejemplo, los dos elementos <div> interiores tienen la misma altura del 50%. Sin embargo, sólo el primer <div> se ve correctamente, ya que sólo el primer elemento contenedor tiene establecida su altura de forma explícita. La altura del segundo <div> se ignora y se sustituye por el valor auto:

div { height: 150px; }

div { height: 50%; }

div { height: auto; }

div { height: 50%; }

Si se establece la altura de un elemento con la unidad de medida em, la referencia que se toma es el tamaño de letra del propio elemento. En el siguiente ejemplo, los dos <div> tienen el mismo valor en la propiedad height, pero como su tamaño de letra es muy diferente, su altura también es muy diferente:

div {

height: 5em;

font-size: 1em;

}

div {

height: 5em;

font-size: 2em;

}

El valor auto es el valor por defecto de la altura de todos los elementos. En este caso, el navegador determina la altura de cada elemento teniendo en cuenta, entre otros, el tipo de elemento que se trata (de bloque o en línea), el sitio disponible en la pantalla del navegador y los contenidos de los elementos.

Por último, recuerda que el valor de la propiedad height siempre hace referencia a la altura del contenido de cada elemento. Si se añaden márgenes, bordes y rellenos, la altura total con la que se muestra el elemento será mayor que el valor de la propiedad height:

div {

height: 100px;

}

div {

height: 100px;

padding-top: 30px;

border: 5px solid #000;

# 12.4. Alturas/anchuras máximas y mínimas

Cuando se diseña la estructura de una página web, se debe tomar la decisión de optar por un diseño de anchura fija o un diseño cuya anchura se adapta a la anchura de la ventana del navegador.

Sin embargo, la mayoría de las veces sería conveniente una solución intermedia: que la anchura de la página sea variable y se adapte a la anchura de la ventana del navegador, pero respetando ciertos límites. En otras palabras, que la anchura de la página no sea tan pequeña como para que no se puedan mostrar correctamente los contenidos y tampoco sea tan ancha como para que las líneas de texto no puedan leerse cómodamente.

CSS define cuatro propiedades que permiten limitar la anchura y altura mínima y máxima de cualquier elemento de la página. Las propiedades son max-width, min-width, max-height y min-height.

| **Propiedad** | **max-width** |
| --- | --- |
| **Valores** | [unidad de medida](https://librosweb.es/referencia/css/unidades_de_medida.html) | [porcentaje](https://librosweb.es/referencia/css/porcentajes.html) | none | [inherit](https://librosweb.es/referencia/css/el_valor_inherit.html) |
| **Se aplica a** | Todos los elementos salvo filas y grupos de filas de tablas |
| **Valor inicial** | none |
| **Descripción** | Permite definir la anchura máxima de un elemento |
| **Propiedad** | **min-width** |
| **Valores** | [unidad de medida](https://librosweb.es/referencia/css/unidades_de_medida.html) | [porcentaje](https://librosweb.es/referencia/css/porcentajes.html) | [inherit](https://librosweb.es/referencia/css/el_valor_inherit.html) |
| **Se aplica a** | Todos los elementos salvo filas y grupos de filas de tablas |
| **Valor inicial** | 0 |
| **Descripción** | Permite definir la anchura mínima de un elemento |

| **Propiedad** | **max-height** |
| --- | --- |
| **Valores** | [unidad de medida](https://librosweb.es/referencia/css/unidades_de_medida.html) | [porcentaje](https://librosweb.es/referencia/css/porcentajes.html) | none | [inherit](https://librosweb.es/referencia/css/el_valor_inherit.html) |
| **Se aplica a** | Todos los elementos salvo columnas y grupos de columnas de tablas |
| **Valor inicial** | none |
| **Descripción** | Permite definir la altura máxima de un elemento |
| **Propiedad** | **min-height** |
| **Valores** | [unidad de medida](https://librosweb.es/referencia/css/unidades_de_medida.html) | [porcentaje](https://librosweb.es/referencia/css/porcentajes.html) | [inherit](https://librosweb.es/referencia/css/el_valor_inherit.html) |
| **Se aplica a** | Todos los elementos salvo columnas y grupos de columnas de tablas |
| **Valor inicial** | 0 |
| **Descripción** | Permite definir la altura mínima de un elemento |

De esta forma, para conseguir un diseño de anchura variable pero controlada, se podrían utilizar reglas CSS como la siguiente:

#contenedor {

min-width: 500px;

max-width: 900px;

}

Las propiedades que definen la altura y anchura máxima y mínima se pueden aplicar a cualquier elemento, aunque solamente suelen utilizarse para estructurar la página. En general, las propiedades más utilizadas son max-width y min-width, ya que no es muy habitual definir alturas máximas y mínimas.

Desafortunadamente, Internet Explorer 6 y las versiones anteriores no soportan ninguna de las cuatro propiedades sobre ningún elemento. Hasta que se incorpore en las nuevas versiones del navegador, es preciso recurrir a trucos que simulen el comportamiento de las propiedades:

max-width equivalente para Internet Explorer:

div {

max-width: 800px;

width: expression(document.body.clientWidth > 801? "800px": "auto");

}

min-width equivalente para Internet Explorer:

div {

min-width:800px;

width: expression(document.body.clientWidth < 801? "800px": "auto" );

}

max-height equivalente para Internet Explorer:

div {

max-height: 300px;

overflow: hidden;

height: expression(this.scrollHeight > 301? "300px" : "auto" );

}

min-height equivalente para Internet Explorer:

div {

min-height:300px;

overflow: hidden;

height: expression(this.scrollHeight < 301? "300px" : "auto" );

}

**Propiedad width**

|  |  |
| --- | --- |
| **Definición** | Establece la anchura de los elementos de bloque |
| **Valores  permitidos** | Uno y sólo uno de los siguientes valores:   * [medida](http://librosweb.es/referencia/css/unidades_de_medida.html) * [porcentaje](http://librosweb.es/referencia/css/porcentajes.html) * auto * [inherit](http://librosweb.es/referencia/css/inherit.html) |
| Si se utilizan porcentajes, hace referencia a la anchura del elemento en el que se encuentra |
| **Valor inicial** | auto |
| **Se aplica a** | Todos los elementos, salvo los elementos en línea que no sean imágenes, las filas de tabla y los grupos de filas de tabla. |
| **Válida en** | [medios visuales](http://librosweb.es/referencia/css/medios.html) |
| **Se hereda** | no |
| **Definición en el estándar** | [w3.org/TR/CSS21/visudet.html#propdef-width](http://www.w3.org/TR/CSS21/visudet.html#propdef-width) |

**Ejemplos de uso**

Para los elementos de bloque y las imágenes, la propiedad width permite establecer su anchura directamente mediante una medida. El siguiente ejemplo establece la anchura a un <div>, de modo que no ocupe todo el sitio disponible:

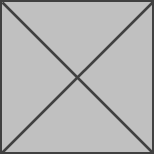
div { width: 150px; }

Aunque ahora el <div> no ocupa toda la anchura disponible, sigue siendo un elemento de bloque, lo que significa que si se coloca seguido otro <div> idéntico, se mostrarán uno debajo de otro:

div { width: 150px; }

div { width: 150px; }

La siguiente imagen tiene un estilo CSS de width: 50px, equivalente a establecer el atributo width de HTML:



La propiedad width no funciona con los elementos en línea que no son imágenes. En el siguiente ejemplo, a pesar de que a cada enlace se le asigna una anchura diferente, el navegador los muestra con la misma anchura:

[(width: 50px) Lorem ipsum dolor sit amet](http://librosweb.es/referencia/css/width.html)   
  
[(width: 150px) Lorem ipsum dolor sit amet](http://librosweb.es/referencia/css/width.html)   
  
[(width: 250px) Lorem ipsum dolor sit amet](http://librosweb.es/referencia/css/width.html)

Si se utilizan unidades de medida, los valores indicados no pueden ser negativos. Si en vez de una unidad de medida se indica un porcentaje, la referencia de ese valor es la anchura del elemento que lo contiene.

En el siguiente ejemplo, los dos elementos <div> interiores tienen la misma anchura del 50%, pero como la anchura de su elemento contenedor es diferente, la anchura real de cada <div> interno también es diferente:

div { width: 50%; }

div { width: 50%; }

Si se establece la anchura de un elemento con la unidad de medida em, el valor indicado toma como referencia el tamaño de letra del propio elemento. En el siguiente ejemplo, los dos <div> tienen el mismo valor en la propiedad width, pero como su tamaño de letra es muy diferente, su anchura en pantalla también es muy diferente:

div { width: 20em; font-size: 1em; }

div { width: 20em; font-size: 2em; }

El valor auto es el valor por defecto de la anchura de todos los elementos. En este caso, el navegador determina la anchura de cada elemento teniendo en cuenta, entre otros, el tipo de elemento que se trata (de bloque o en línea), el sitio disponible en la pantalla del navegador y los contenidos de los elementos.

Por último, recuerda que el valor de la propiedad width siempre hace referencia a la anchura del contenido de cada elemento. Si se añaden márgenes, bordes y rellenos, la anchura total con la que se muestra el elemento será mayor que el valor de la propiedad width:

div { width: 350px; }

div { width: 350px; padding: 30px; border: 5px solid #000; }

**Propiedad margin**

|  |  |
| --- | --- |
| **Definición** | Establece la anchura de algunos o todos los márgenes de los elementos |
| **Valores  permitidos** | Uno, dos, tres o cuatro de los siguientes valores:   * [medida](https://librosweb.es/referencia/css/unidades_de_medida.html) * [porcentaje](https://librosweb.es/referencia/css/porcentajes.html) * auto * [inherit](https://librosweb.es/referencia/css/inherit.html) |
| Si se utilizan porcentajes, hace referencia a la anchura del elemento en el que se encuentra |
| **Valor inicial** | Cada margen define su propio valor por defecto |
| **Se aplica a** | Todos los elementos |
| **Válida en** | [medios visuales](https://librosweb.es/referencia/css/medios.html) |
| **Se hereda** | no |
| **Definición en el estándar** | [w3.org/TR/CSS21/box.html#propdef-margin](http://www.w3.org/TR/CSS21/box.html#propdef-margin) |

**Ejemplos de uso**

La propiedad margin es una de las *"propiedades shorthand"* que define CSS y que se utilizan para establecer de forma abreviada el valor de una o más propiedades individuales.

Si se quiere establecer el mismo valor para todos los márgenes de un elemento, se podría utilizar la siguiente regla CSS:

div {

margin-top: 10px;

margin-right: 10px;

margin-bottom: 10px;

margin-left: 10px;

}

Utilizando la propiedad margin, es posible establecer de forma abreviada el mismo valor a todos los márgenes de un elemento:

div {

margin: 10px;

}

La propiedad margin es tan poderosa que permite establecer uno, dos, tres o los cuatro márgenes de forma simultánea.

Si se indican dos valores, el primero hace referencia a los márgenes verticales y el segundo es el valor de los márgenes horizontales, de modo que las siguientes reglas CSS son equivalentes:

div {

margin: 10px 20px;

}

div {

margin-top: 10px;

margin-right: 20px;

margin-bottom: 10px;

margin-left: 20px;

}

Si se indican tres valores, el primero hace referencia al margen superior, el segundo es el valor de los márgenes horizontales y el tercero es el margen inferior, de modo que las siguientes reglas CSS son equivalentes:

div {

margin: 10px 20px 30px;

}

div {

margin-top: 10px;

margin-right: 20px;

margin-bottom: 30px;

margin-left: 20px;

}

Por último, si se indican cuatro valores, el primero es el margen superior, el segundo es el margen derecho, el tercero es el margen inferior y el cuarto es el margen izquierdo, de modo que las siguientes reglas CSS son equivalentes:

div {

margin: 10px 20px 30px 40px;

}

div {

margin-top: 10px;

margin-right: 20px;

margin-bottom: 30px;

margin-left: 40px;

}

El uso de propiedades como margin no sólo hace más breve el código CSS, sino que demuestra que se tienen conocimientos avanzados de CSS.

**Propiedad margin-bottom**

|  |  |
| --- | --- |
| **Definición** | Establece la anchura del margen inferior de los elementos de bloque |
| **Valores  permitidos** | Uno y sólo uno de los siguientes valores:   * [medida](http://librosweb.es/referencia/css/unidades_de_medida.html) * [porcentaje](http://librosweb.es/referencia/css/porcentajes.html) * auto * [inherit](http://librosweb.es/referencia/css/inherit.html) |
| Si se utilizan porcentajes, hace referencia a la anchura del elemento en el que se encuentra |
| **Valor inicial** | 0 |
| **Se aplica a** | Todos los elementos, salvo los elementos en línea que no sean imágenes |
| **Válida en** | [medios visuales](http://librosweb.es/referencia/css/medios.html) |
| **Se hereda** | no |
| **Definición en el estándar** | [w3.org/TR/CSS21/box.html#propdef-margin-bottom](http://www.w3.org/TR/CSS21/box.html#propdef-margin-bottom) |

**Ejemplos de uso**

Para los elementos de bloque y las imágenes, la propiedad margin-bottom establece la anchura de su margen inferior, como se muestra en el siguiente ejemplo:

div { margin-bottom: 20px; }

También es posible utilizar valores negativos en el margen inferior. En el siguiente ejemplo, el <div> tiene un margen inferior de -20px, lo que en la práctica implica que el elemento *"se sale"* por la parte inferior de su elemento contenedor:

div { margin-bottom: -20px; }

Si se utilizan porcentajes, su referencia es la anchura del elemento contenedor. Si no se ha establecido de forma explícita una anchura, se sustituye por el valor auto.

El aspecto más importante relacionado con los márgenes es que CSS fusiona de forma automática los márgenes verticales de los elementos de bloque.

Si en una página web dos elementos de bloque contiguos definen sus propios márgenes verticales (mediante las propiedades margin-top y margin-bottom), el navegador los fusiona de forma automática y muestra un margen vertical igual al mayor de los dos márgenes verticales.

En el siguiente ejemplo, el primer <div> define un margen inferior de 40px, mientras que el segundo <div> define un margen superior de 20px. Como se trata de dos márgenes verticales contiguos, el navegador los fusiona y se queda con el más grande (en este caso, los 40px):

div { margin-bottom: 40px; }

div { margin-top: 20px; }

La fusión automática de márgenes verticales no se realiza en los elementos posicionados mediante la propiedad [float](http://librosweb.es/referencia/css/float.html) y en los elementos posicionados de forma absoluta (position: absolute).

**PROPIEDADES RELACIONADAS**

* [margin-right](http://librosweb.es/referencia/css/margin-right.html)
* [margin-left](http://librosweb.es/referencia/css/margin-left.html)
* [margin-top](http://librosweb.es/referencia/css/margin-top.html)
* [margin-bottom](http://librosweb.es/referencia/css/margin-bottom.html)
* [margin](http://librosweb.es/referencia/css/margin.html)

© 2006-2016 LibrosWeb.es [Contacto](http://librosweb.es/sitio/contacto) [Novedades](https://plus.google.com/+librosweb) [Condiciones](http://librosweb.es/sitio/condiciones) [Privacidad](http://librosweb.es/sitio/privacidad)