Contenido

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

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

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

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

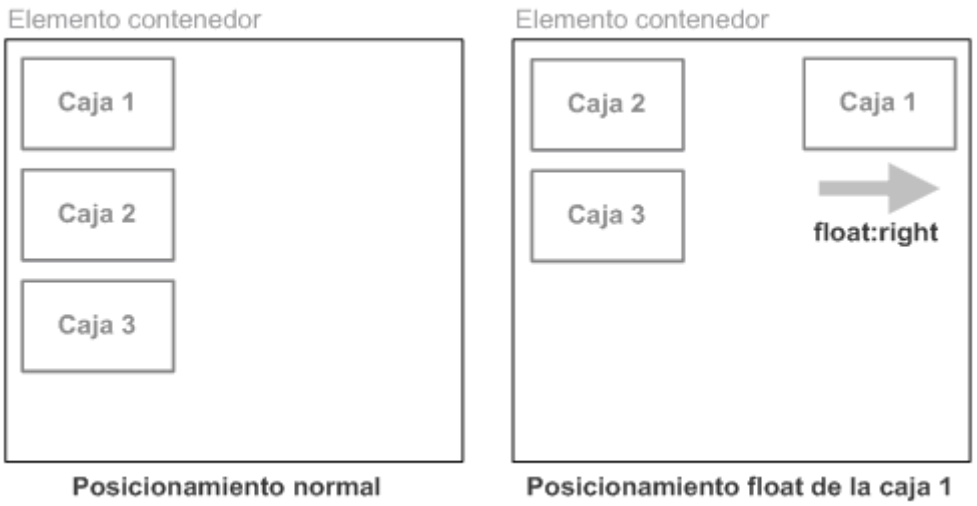
El posicionamiento de una caja se establece mediante la propiedad *position.*

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 están diseñadas con el posicionamiento flotante.

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.

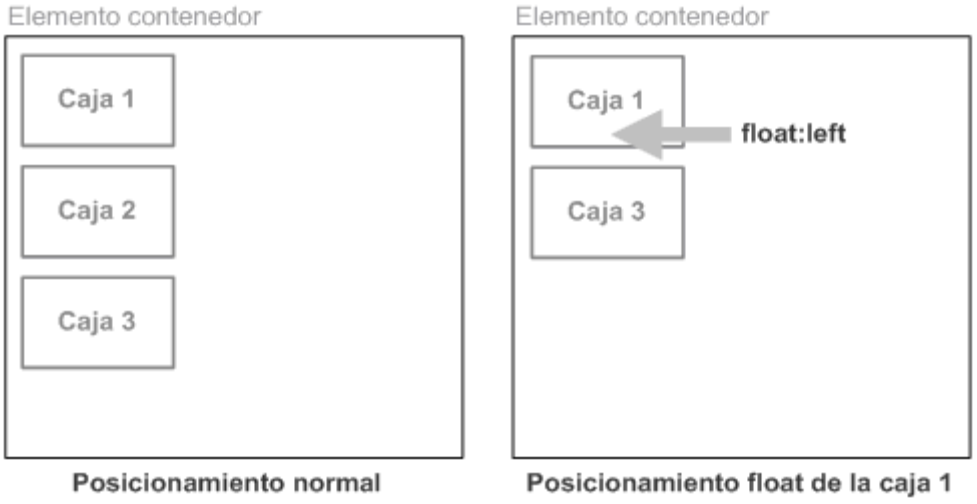
La siguiente imagen muestra **el resultado de posicionar de forma flotante hacia la derecha la caja 1**:



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.

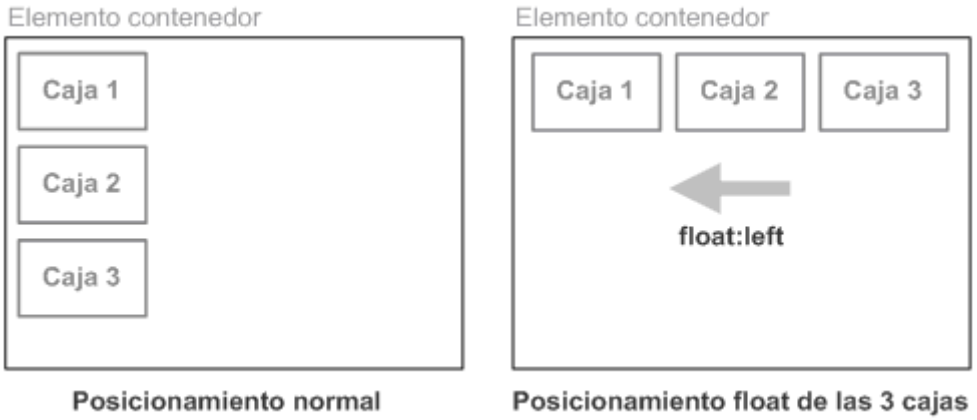
Si ahora **posicionamos la caja 1, de forma flotante, hacia la izquierda**, el resultado es el que muestra la siguiente imagen:



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 dónde estaba la caja 1 y la caja 3 se muestra dónde estaba la caja 2.

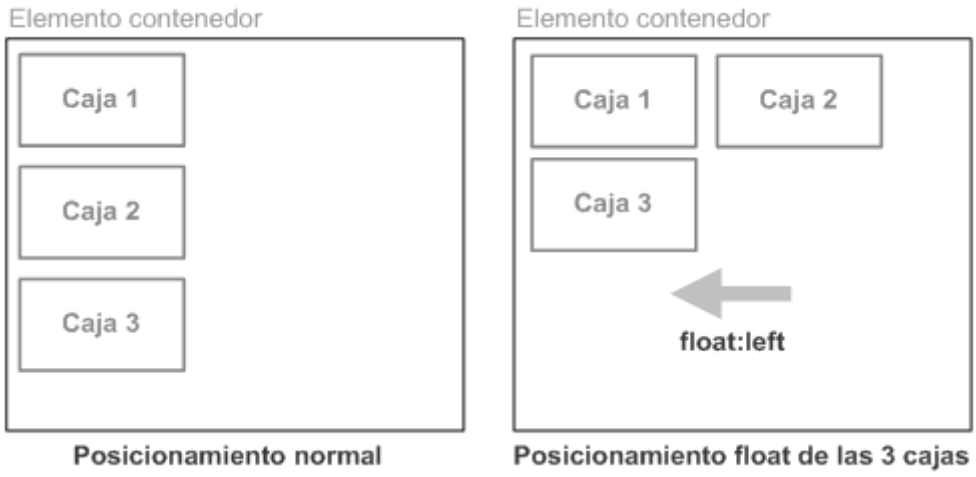
Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. **El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos**.

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



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

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

Propiedad	**float**

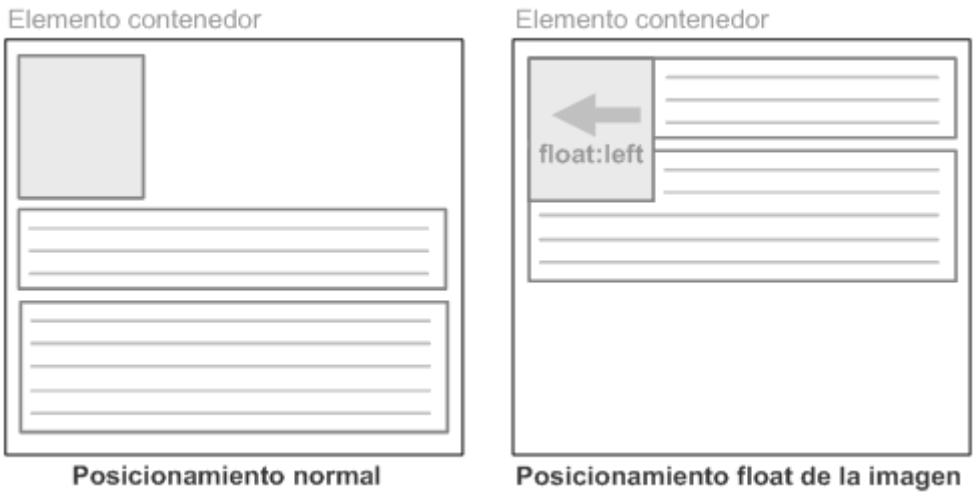
Valores	left | right | none | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el tipo de posicionamiento flotante del elemento

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

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

Propiedad clear

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



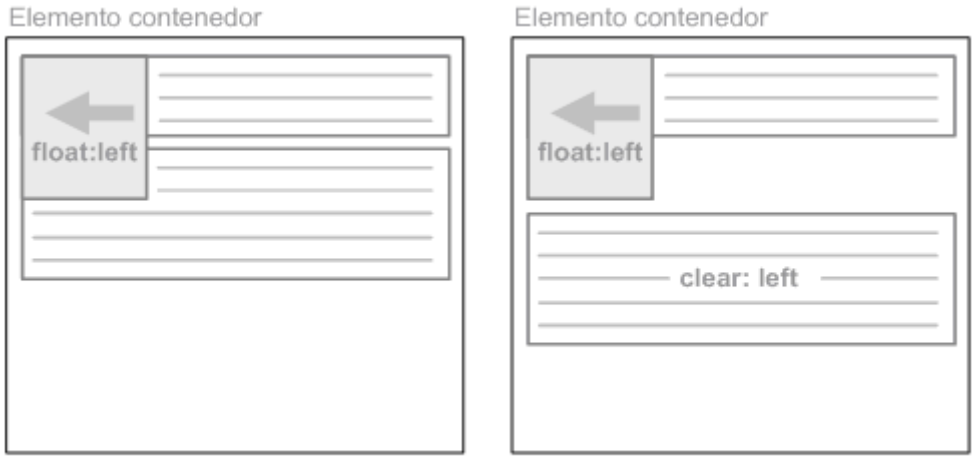
La regla CSS que se aplica en la imagen del ejemplo anterior es:

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

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

CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante *float*. De hecho, **en muchas ocasiones es admisible que algunos**

contenidos fluyan alrededor de una imagen, *pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen*:



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

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

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

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

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

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

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

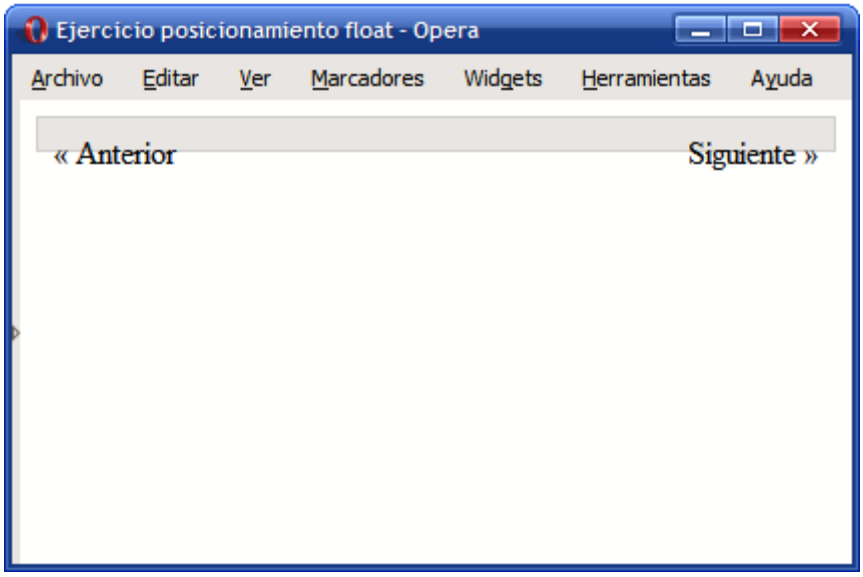
El valor *both* despeja los lados izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante hacia la izquierda o hacia la derecha.

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:



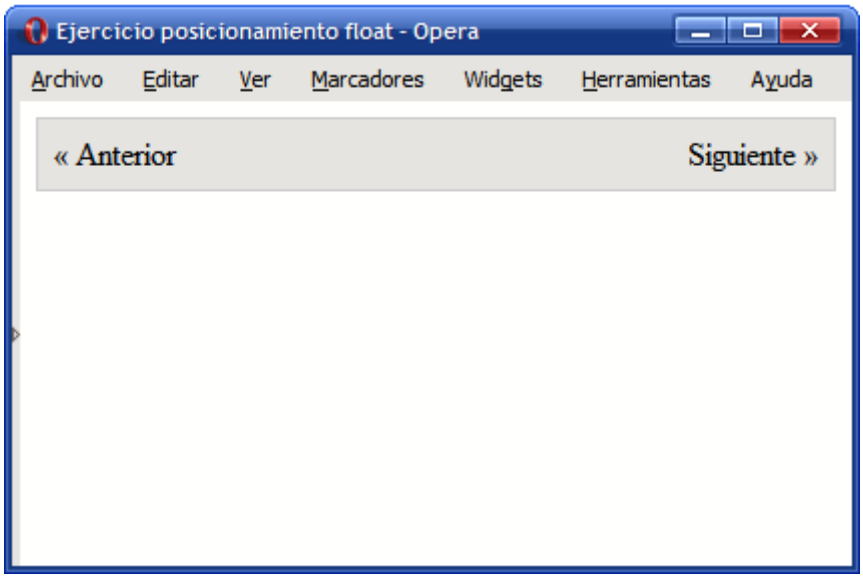
Los elementos Anterior y Siguiente *se salen* de su elemento contenedor y el resultado es visualmente incorrecto. El motivo de este comportamiento es que un elemento posicionado de forma flotante ya no pertenece al flujo normal de la página HTML. Por tanto, el elemento en realidad no encierra ningún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

La solución consiste en utilizar la propiedad ***overflow*** (luego se comenta) 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:



Visualización

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

Utilizando algunas de estas propiedades es posible ocultar y/o hacer invisibles las cajas de los elementos.

Propiedades display y visibility

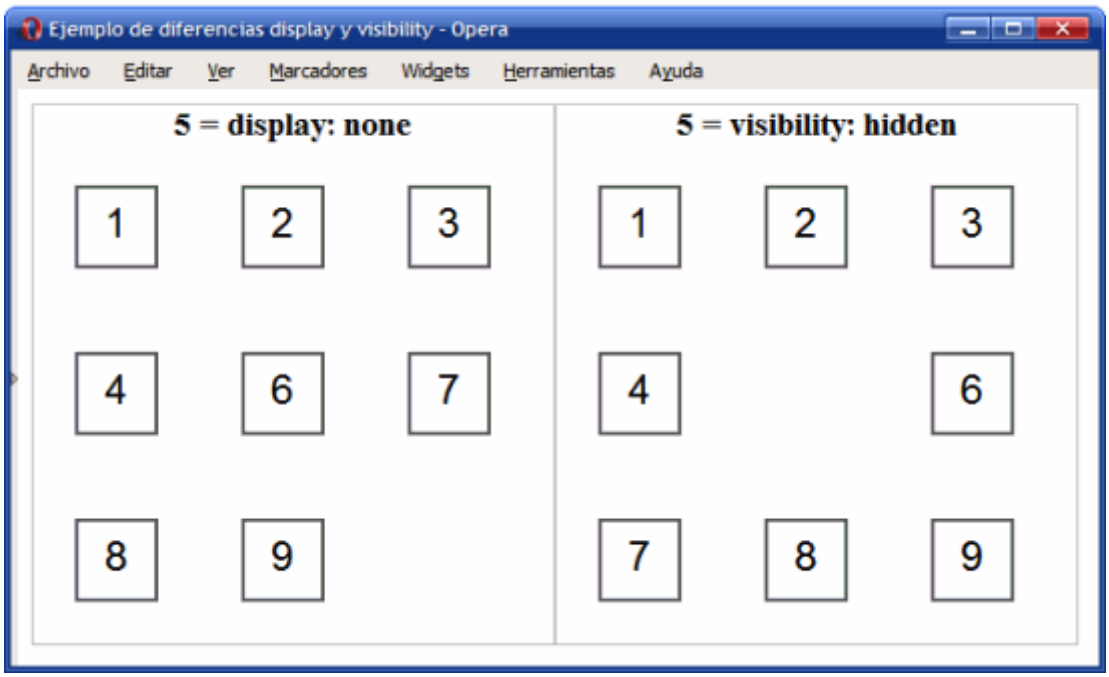
Las propiedades *display* y *visibility* controlan la visualización de los elementos. Las dos propiedades permiten ocultar cualquier elemento de la página.

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

Por otra parte, **la propiedad *visibility* permite hacer invisible un elemento, lo que significa que el navegador crea la caja del elemento pero no la muestra. En este caso, el resto de**

elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

La siguiente imagen muestra la diferencia entre ocultar la caja número 5 mediante la propiedad *display* o hacerla invisible mediante la propiedad *visibility*:



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

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

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

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

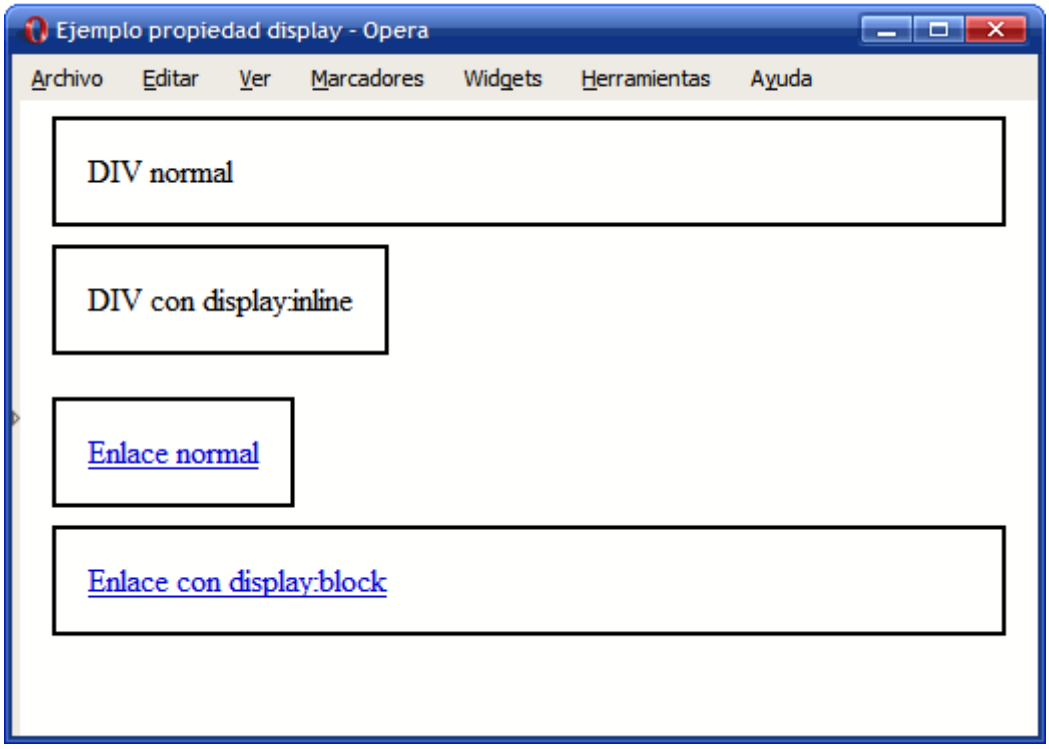
Los valores más utilizados son ***inline*, *block* y *none***. El valor ***block* muestra un elemento como si fuera un elemento de bloque, independientemente del tipo de elemento que se trate**. El valor ***inline* visualiza un elemento en forma de elemento en línea, independientemente del tipo de elemento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

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

http://www.w3schools.com/cssref/pr_class_display.asp

http://www.w3schools.com/cssref/playit.asp?filename=playcss_display&preval=inline

El siguiente ejemplo muestra el uso de la propiedad display para mostrar un elemento de bloque como si fuera un elemento en línea y para mostrar un elemento en línea como si fuera un elemento de bloque:



Las reglas CSS del ejemplo anterior son las siguientes:

```
<div>DIV normal</div>
<div style="display:inline">DIV con display:inline</div>
```

```
<a href="#">Enlace normal</a>
<a href="#" style="display:block">Enlace con display:block</a>
```

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

La definición completa de la propiedad *visibility* es mucho más sencilla:

Propiedad	**visibility**
Valores	visible | hidden | collapse | inherit
Se aplica a	Todos los elementos
Valor inicial	visible
Descripción	Permite hacer visibles e invisibles a los elementos

Las posibilidades de la propiedad *visibility* son mucho más limitadas que las de la propiedad *display*, ya que sólo permite hacer visibles o invisibles a los elementos de la página.

Inicialmente todas las cajas que componen la página son visibles. **Empleando el valor *hidden* es posible convertir una caja en invisible para que no muestre sus contenidos. El resto de elementos de la página se muestran como si la caja todavía fuera visible, por lo que en el lugar donde originalmente se mostraba la caja invisible, ahora se muestra un hueco vacío**.

Por último, el valor *collapse* de la propiedad *visibility* sólo se puede utilizar en las filas, grupos de filas, columnas y grupos de columnas de una tabla. Su efecto es similar al de la propiedad *display*, ya que oculta completamente la fila y/o columna y se pueden mostrar otros contenidos en ese lugar. Si se utiliza el valor *collapse* sobre cualquier otro tipo de elemento, su efecto es idéntico al valor *hidden*.

Relación entre display, float y position

Cuando se establecen las propiedades *display*, *float* y *position* sobre una misma caja, su interpretación es la siguiente:

- **Si *display* vale *none*, se ignoran las propiedades *float* y position y la caja no se muestra en la página.**

- **Si position vale *absolute* o *fixed*, la caja se posiciona de forma absoluta, se considera que *float* vale *none* y la propiedad *display* vale *block* tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades top, right, bottom y left.**

- **En cualquier otro caso, si *float* tiene un valor distinto de *none*, la caja se posiciona de forma flotante y la propiedad *display* vale *block* tanto para los elementos en línea como para los elementos de bloque.**

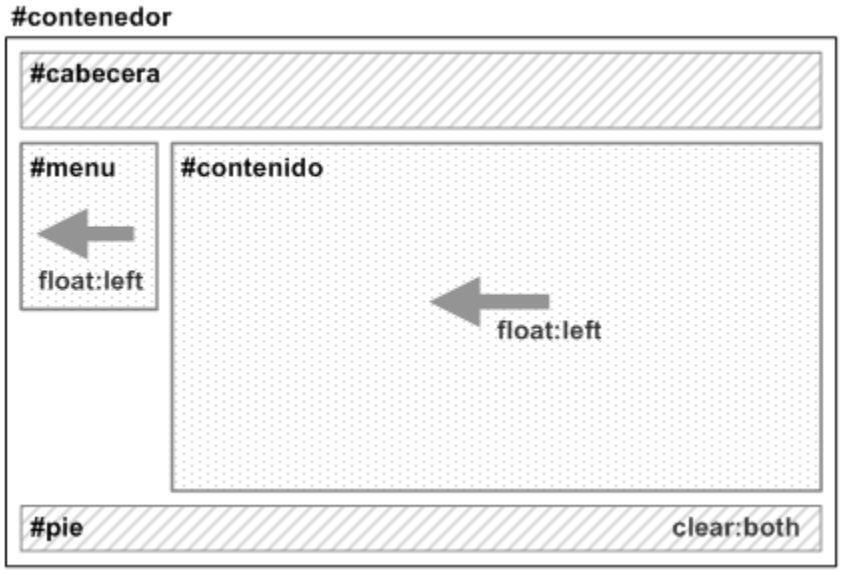
Estructura o layout

Diseño a 2 columnas con cabecera y pie de página

El objetivo de este diseño es definir una estructura de página con cabecera y pie, un menú lateral de navegación y una zona de contenidos. La anchura del menú y de los contenidos puede ser de un 20% y un 80% respectivamente.

La solución CSS se basa en el uso de la propiedad *float* para los elementos posicionados como el menú y los contenidos y el uso de la propiedad *clear* en el pie de página para evitar los solapamientos ocasionados por los elementos posicionados con *float*.

Las propiedades CSS necesarias en el diseño a dos columnas con cabecera y pie de página serán:



El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

(La anchura de la página se fija en 700px, la anchura del menú es de 150px y la anchura de los contenidos es de 550px)

```
#contenedor {
width: 700px;
}
#cabecera {
}
#menu {
float: left;
width: 150px;
}
#contenido {
float: left;
width: 550px;
}
#pie {
clear: both;
}
```

```
<body>
<div id="contenedor">
<div id="cabecera">
</div>

<div id="menu">
</div>

<div id="contenido">
</div>

<div id="pie">
</div>
</div>
</body>
```

En los estilos CSS anteriores se ha optado por desplazar tanto el menú como los contenidos hacia la izquierda de la página (float:left). Sin embargo, en este caso también se podría desplazar el menú hacia la izquierda (float:left) y los contenidos hacia la derecha (float: right).

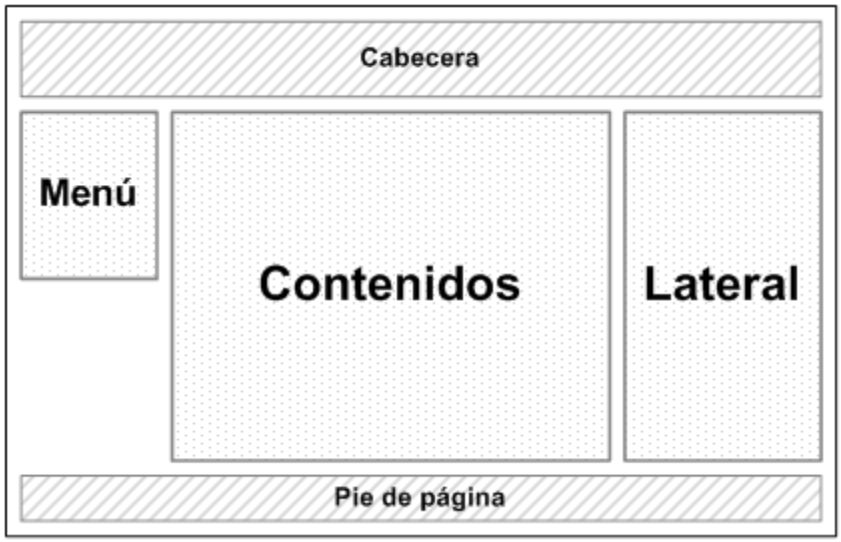
El diseño anterior es de anchura fija, lo que significa que no se adapta a la anchura de la ventana del navegador. Para conseguir una página de anchura variable y que se adapte de forma dinámica a la ventana del navegador, se deben aplicar las siguientes reglas CSS:

```
#contenedor {
}
#cabecera {
}
#menu {
float: left;
width: 15%;
}
#contenido {
float: left;
width: 85%;
}
#pie {
clear: both;
}
```

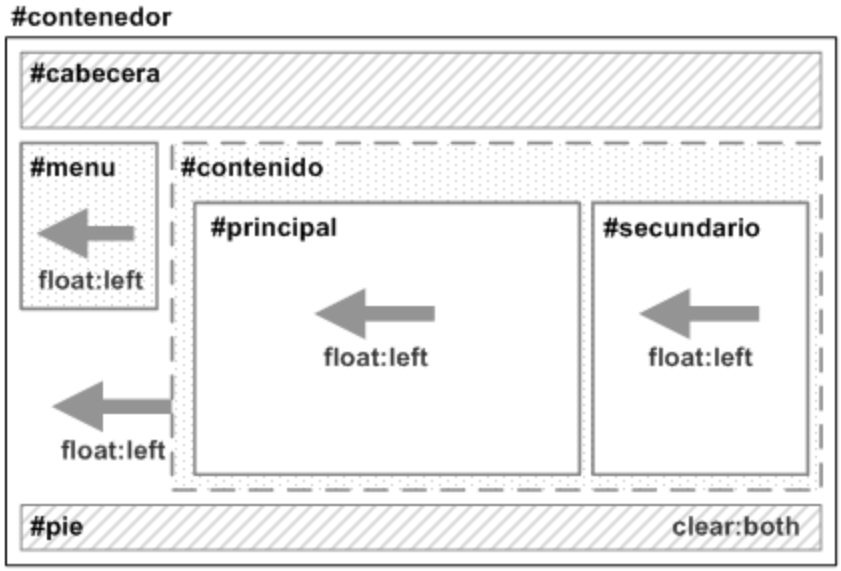
Si se indican las anchuras de los bloques que forman la página en porcentajes, el diseño final es dinámico.

Diseño a 3 columnas con cabecera y pie de página

Además del diseño a dos columnas, el diseño más utilizado es el de tres columnas con cabecera y pie de página. En este caso, los contenidos se dividen en dos zonas diferenciadas: zona principal de contenidos y zona lateral de contenidos auxiliares:



La solución CSS emplea la misma estrategia del diseño a dos columnas y se basa en utilizar las propiedades *float* y *clear*:



El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

```
#contenedor {
}
#cabecera {
}
#menu {
float: left;
width: 15%;
}
#contenido {
float: left;
width: 85%;
}
#contenido #principal {
float: left;
width: 80%;
}
#contenido #secundario {
float: left;
width: 20%;
}
#pie {
clear: both;
}
```

```
<body>
<div id="contenedor">
<div id="cabecera">
</div>

<div id="menu">
</div>

<div id="contenido">
<div id="principal">
</div>

<div id="secundario">
</div>
</div>

<div id="pie">
</div>
</div>
</body>
```

El código anterior crea una página con anchura variable que se adapta a la ventana del navegador.

Al igual que sucedía en el diseño a dos columnas, se puede optar por posicionar todos los elementos mediante float: left o se puede utilizar float: left para el menú y la zona principal de contenidos y float: right para el contenedor de los contenidos y la zona secundaria de contenidos

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 | porcentaje | none | inherit
Se aplica a	Todos los elementos salvo filas y grupos de filas de tablas
Valor inicial	none
Descripción	Permite definir la anchura máxima de un elemento

Propiedad	**min-width**
Valores	unidad de medida | porcentaje | inherit

Se aplica a	Todos los elementos salvo filas y grupos de filas de tablas
Valor inicial	0
Descripción	Permite definir la anchura mínima de un elemento

Propiedad	**max-height**
Valores	unidad de medida | porcentaje | none | inherit
Se aplica a	Todos los elementos salvo columnas y grupos de columnas de tablas
Valor inicial	none
Descripción	Permite definir la altura máxima de un elemento

Propiedad	**min-height**
Valores	unidad de medida | porcentaje | inherit
Se aplica a	Todos los elementos salvo columnas y grupos de columnas de tablas
Valor inicial	0
Descripción	Permite definir la altura mínima de un elemento

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

```
#contenedor {
min-width: 500px;
```

```
max-width: 900px;
}
```

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

Centrar una página horizontalmente

A medida que aumenta el tamaño y la resolución de las pantallas de ordenador, se hace más difícil diseñar páginas que se adapten al tamaño de la ventana del navegador. El principal reto que se presenta con resoluciones superiores a 1024 x 768 píxel, es que las líneas de texto son demasiado largas como para leerlas con comodidad. Por ese motivo, normalmente se opta por diseños con una anchura fija limitada a un valor aceptable para mantener la legibilidad del texto.

Por otra parte, los navegadores alinean por defecto las páginas web a la izquierda de la ventana. Cuando la resolución de la pantalla es muy grande, la mayoría de páginas de anchura fija alineadas a la izquierda parecen muy estrechas y provocan una sensación de vacío.

La solución más sencilla para evitar los grandes espacios en blanco consiste en crear páginas con una anchura fija adecuada y centrar la página horizontalmente respecto de la ventana del navegador.

Utilizando la propiedad *margin*de CSS, es muy sencillo centrar una página web horizontalmente. La solución consiste en agrupar todos los contenidos de la página en un elemento y asignarle a ese unos márgenes laterales automáticos. El que encierra los contenidos se suele llamar *contenedor* (en inglés se denomina wrapper o container):

```
#contenedor {
width: 300px;
margin: 0 auto;
```

```
}
<body>
<div id="contenedor">
<h1>Loremipsum dolor sitamet</h1>
...
</div>
</body>
```

El valor *0 auto* significa que los márgenes superior e inferior son iguales a 0 y los márgenes laterales toman un valor de *auto*. Cuando se asignan márgenes laterales automáticos a un elemento, los navegadores centran ese elemento respecto de su elemento padre. En este ejemplo, el elemento padre del es la propia página (el elemento <body>), por lo que se consigue centrar el elemento respecto de la ventana del navegador.

Modificando ligeramente el código CSS anterior se puede conseguir un diseño dinámico o *líquido*(también llamado *fluido*) que se adapta a la anchura de la ventana del navegador y permanece siempre centrado:

```
#contenedor {
width: 70%;
margin: 0 auto; }
```

Estableciendo la anchura del elemento contenedor mediante un porcentaje, su anchura se adapta de forma continua a la anchura de la ventana del navegador. De esta forma, si se reduce la anchura de la ventana del navegador, la página se verá más estrecha y si se maximiza la ventana del navegador, la página se verá más ancha.

Sitios consultados para realizar este recopilatorio:

http://es.wikipedia.org/

http://www.desarrolloweb.com

http://www.librosweb.es

http://www.w3schools.com/html/