

## Maquetar páginas web con DIVs (capas o layouts)

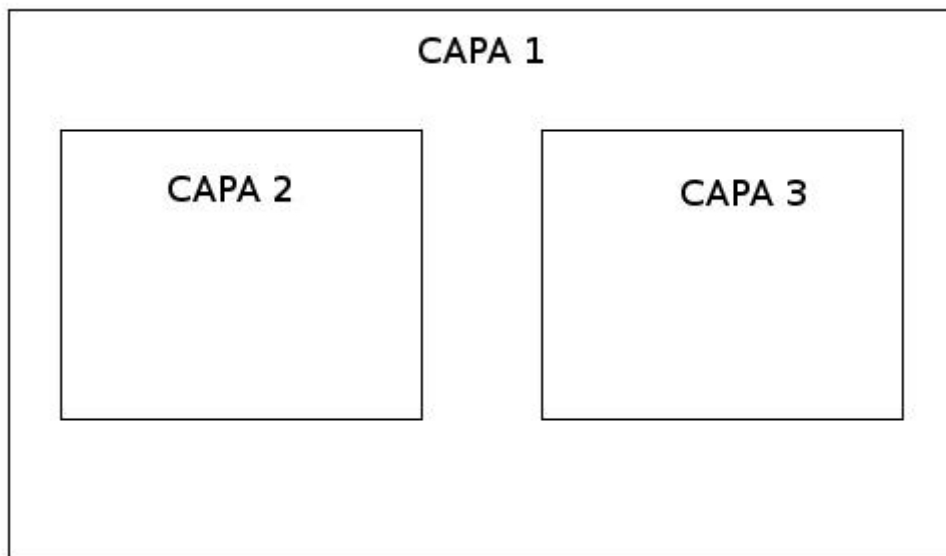
En pocas palabras, *maquetar una página web es pasar el diseño a código HTML*, poniendo cada cosa en su lugar (una cabecera, un menú, etc.).

Hasta hace unos años la única manera de maquetar una página web era mediante tablas HTML (<table>), o frames, pero esto tiene muchas desventajas y limitaciones, por eso la técnica de maquetación fue evolucionando con los años hasta llegar al punto donde no se usan tablas, si no capas (los famosos DIVs) a las que se le dan formato mediante CSS.

### ¿Capas, layouts, divs? ¿que es eso?

Las capas, layouts o divs son la misma cosa con distinto nombre, para tener un concepto mental más fuerte de lo que son, podemos imaginarlos como *contenedores o bloques donde podemos meter lo que queramos dentro* (imágenes, texto, animaciones, otro bloque, o todo al mismo tiempo) a los que se le asigna un ancho, alto y posición, de esta manera se van a ir posicionando consiguiendo la estructura que queremos.

Veamos un ejemplo gráfico:



Como podemos ver, tenemos 3 capas estructuradas de la siguiente manera:

- **Capa 1:** es la capa principal y contenedora
- **Capa 2:** capa dentro de la capa contenedora 1 y alineada a la izquierda (float:left;)
- **Capa 3:** igual que la capa 2, solo que tiene un margen con respecto a esta última (float:left; margin: 10px; ya veremos esto más en detalle).

## Creando nuestro primer DIV

La forma de crear una capa DIV es así:

```
<div></div>
```

recuerda que todo contenido visible de la página debe ir entre las etiquetas

```
<body></body>
```

con el código HTML completo quedaría así:

```
<html>
  <head>
    <title>Curso de maquetacion CSS</title>
  </head>
  <body>
    <div>¡Esta es mi primer capa!</div>
  </body>
</html>
```

## Como darle formato a un DIV con estilos CSS

Para darle formato a un DIV tenemos que identificarlo de alguna forma, para esto existe **el atributo ID**, en el pondremos el nombre del DIV para luego llamarlo desde la hoja de estilos, la forma de escribirlo es así:

```
<div id="capa1">¡Esta es mi primer capa!</div>
```

Como podemos ver, a esta capa le pusimos "capa1" de nombre, ahora solo nos falta abrir la hoja de estilos que creamos y llamarlo de esta manera:

```
#capa1{
  background-color:green;
}
```

esto hará que el color de fondo *de esa capa* sea verde, en la siguiente imagen se puede apreciar:

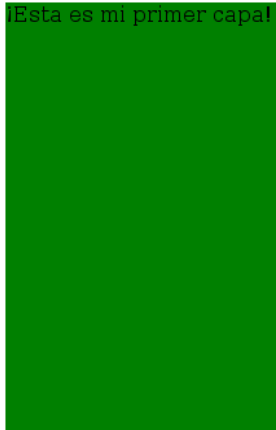


Como vemos, el color se estira, esto es porque no le definimos un ancho determinado, para hacer agregaremos el atributo **width** a #capa1 de la siguiente forma:

```
#capa1{
  width:210px;
  background-color:green;
}
```

Simple, le agregaremos el atributo **height** con el valor en pixeles que queramos, de esta forma:

```
#capa1{
  width:210px;
  height:300px;
  background-color:green;
}
```



Con esto ya tendremos una especie de rectángulo verde donde podremos agregar texto como queremos.

## Flotar y acomodar un DIV

La maquetación por divs CSS se basa en "**flotar**" unas capas dentro de otras para conseguir la estructura que queremos, para esto se utiliza la propiedad *float*.

Esta propiedad se utiliza para flotar los bloques, podemos utilizarla con los siguientes valores:

- **none:** no flota la capa.
- **left:** flota la capa hacia la izquierda.
- **right:** flota la capa hacia la derecha.

Siguiendo con el ejemplo de la lección anterior, vamos a crear otra capa #capa2, le vamos a aplicar otro color de fondo con el mismo alto y ancho pero vamos a flotar ambas capas hacia la izquierda, quedaría así:

```
#capa1{
  width:210px;
  height:300px;
  background-color:green;
  float:left;
}
```

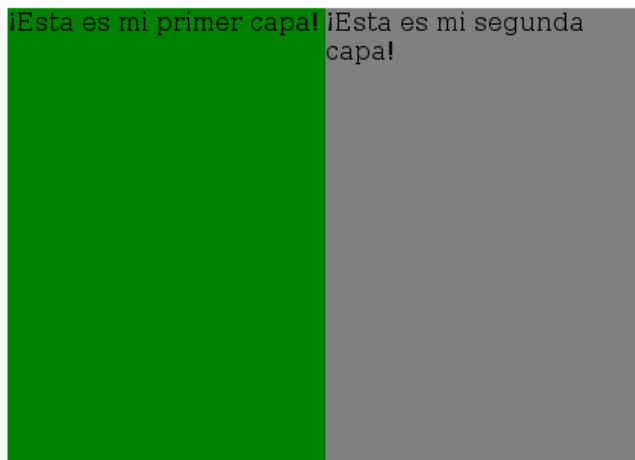
```
#capa2{
  width:210px;
  height:300px;
  background-color:gray;
  float:left;
}
```

y el código HTML sería este:

```
<html>
  <head>
    <title>Curso de maquetacion CSS</title>
    <link href="estilos.css" type="text/css" rel="stylesheet">
  </head>
  <body>
    <div id="capa1">¡Esta es mi primer capa!</div>
    <div id="capa2">¡Esta es mi segunda capa!</div>
  </body>
</html>
```

**Nota:** asegúrate de escribir bien los ID, tanto en la hoja de estilos como en el código HTML, que coincidan en ambos, de lo contrario no funcionará.

Esto se debería ver mas o menos así:



Ahora, ¿que pasa si aplicamos un margen izquierdo a la capa2?

```
#capa2{
  width:210px;
  height:300px;
  background-color:gray;
  float:left;
  margin-left:10px;
}
```

Se debería ver así:



## Evitar que un DIV o capa se superponga con otra

Seguramente cuando estés maquetando tu sitio web necesitarás tener una capa o bloque que no tenga capas a su/s lados, para eso está la propiedad CSS *Clear*.

Esta propiedad se utiliza en conjunto con *float* y **sirve para evitar que una capa se posicione a cualquiera de los lados**, los valores que admite son estos:

- **left:** no deja que una capa flote a la izquierda
- **right:** evita que una capa flote a la derecha
- **both:** evita que haya capas flotantes en cualquiera de sus lados

Nosotros lo usaremos para crear, por ejemplo, el pie de página.

Siguiendo con el ejemplo que hicimos en la lección anterior vamos a crear una tercer capa #capa3 y le aplicaremos esta propiedad, añadiremos un ancho de 430 píxeles y un alto de 30 píxeles, también le aplicaremos un color de fondo naranja.

```
#capa3{  
    width:430px;  
    height:30px;  
    background-color:orange;  
    float:left;  
    clear:both;  
}
```

el código HTML quedaría así:

```
<html>  
  <head>
```

```
<title>Curso de maquetacion CSS</title>
<link href="estilos.css" type="text/css" rel="stylesheet">
</head>
<body>
  <div id="capa1">¡Esta es mi primer capa!</div>
  <div id="capa2">¡Esta es mi segunda capa!</div>
  <div id="capa3">¡Esta es mi tercer capa!</div>
</body>
</html>
```

Esto dará como resultado algo parecido a esto:



Ahora vamos a agregar un margen superior para separar un poco el pie de pagina (#capa3) de las demás capas o divs.

```
#capa3{
  width:430px;
  height:30px;
  background-color:orange;
  float:left;
  clear:both;
  margin-top:10px;
}
```

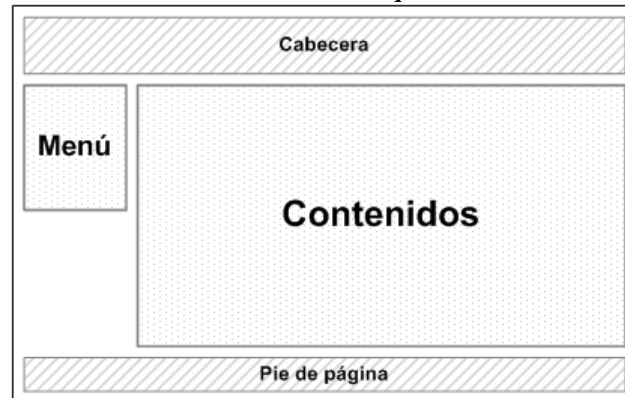
Esto quedaría así:



## Vamos a maquetar una página con la siguiente estructura:

- **Cabecera:** aquí puede ir un logo o el nombre de tu página web
- **Barra lateral:** podemos poner un menú vertical con listas HTML
- **Contenido:** donde irá el contenido de la web
- **Pie de página:** podemos poner lo que necesitemos, desde el copyright hasta los enlaces recomendados

La estructura deberá quedar así:



## Maquetar el contenedor general con CSS

La primera capa que tendremos que crear es un contenedor general al que le **aplicaremos un ancho y alto fijo** para todo el sitio Web.

Para este ejemplo lo haremos con estas medidas:

- **Ancho:** 900 pixeles
- **Alto:** 400 pixeles

Sabiendo esto podemos ponernos manos a la obra :)

A este DIV lo llamaremos "contenedor", y sería algo así:

```
#contenedor {  
    width: 900px;  
    height: 400px;  
}
```

Ahora abrimos el archivo HTML (index.html) y comenzamos a armar las capas, por ahora solo tenemos que llamar a la capa "contenedor", así:

```
<html>  
<head>  
<title>Maquetando con estilos CSS</title>
```

```
<link href="estilos.css" rel="stylesheet" type="text/css">
</head>
<body>
    <div id="contenedor">

    </div>
</body>
</html>
```

Ya tenemos el contenedor general, ahora maquetaremos la cabecera.

## Maquetar cabecera con CSS

Teniendo el contenedor general y siguiendo el diagrama de maquetación que vimos antes, es hora de crear la cabecera de nuestro sitio, a esta le asignaremos el alto que queremos que tenga, siempre dentro de los límites que definimos en el contenedor general, obviamente usaremos mucho menos, solo 50 pixeles, y también un color de fondo (verde) para ir notando cada capa, quedando así:

```
#cabecera {
    background-color: green;
    height:50px;
    padding:5px;
}
```

y llamamos a esta capa desde el código HTML, pero ojo, tiene que ir dentro de la capa contenedora, así:

```
<html>
<head>
<title>Maquetando con estilos CSS</title>
<link href="estilos.css" rel="stylesheet" type="text/css">
</head>
<body>
    <div id="contenedor">
        <div id="cabecera">Esta es la cabecera</div>
    </div>
</body>
</html>
```

## Maquetar la barra lateral con CSS

Llegó la hora de crear la capa de la barra lateral, donde pondremos un menú o lo que necesitemos. En esta capa, además de agregar otro color de fondo, comenzaremos a utilizar la propiedad float para flotar la barra a la izquierda, además de establecer un ancho y alto que utilizará la capa.

```
#barra-lateral {
    background-color: orange;
    float: left;
    width:140px;
    height:100%;
}
```



Si no queremos que ocupe todo el alto de la página solo ponemos una cantidad de pixeles determinada, sin porcentajes.

El HTML quedará así:

```
<html>
<head>
<title>Maquetando con estilos CSS</title>
<link href="estilos.css" rel="stylesheet" type="text/css">
</head>
<body>
  <div id="contenedor">
    <div id="cabecera">Esta es la cabecera</div>
    <div id="barra-lateral">Esta es la barra lateral</div>
  </div>
</body>
</html>
```

Ya nos falta muy poco...

## Maquetar contenido con CSS

Al igual que la capa de la barra lateral, definiremos los mismos atributos solo que cambiaremos algunos valores como el ancho (el contenido ocupa más lugar) y el color de fondo (para distinguir donde empieza y termina la capa).

```
#contenido {
  background-color: gray;
  float:left;
  width:740px;
  height:100%;
}
```

y en el HTML agregamos la capa debajo de la barra lateral:

```
<html>
<head>
<title>Maquetando con estilos CSS</title>
<link href="estilos.css" rel="stylesheet" type="text/css">
</head>
<body>
  <div id="contenedor">
    <div id="cabecera">Esta es la cabecera</div>
    <div id="barra-lateral">Esta es la barra lateral</div>
    <div id="contenido">Este es el contenido principal</div>
  </div>
</body>
</html>
```

Ya solo nos queda maquetar el pie de página y es hora de que le des tus toques finales.

## Maquetar pie de página CSS

En el pie de página utilizaremos la propiedad `clear` que vimos en la lección anterior, también definiremos un color de fondo:

```
#pie {  
    background-color: blue;  
    clear: both;  
}
```

Finalmente el HTML quedaría así:

```
<html>  
<head>  
<title>Maquetando con estilos CSS</title>  
<link href="estilos.css" rel="stylesheet" type="text/css">  
</head>  
<body>  
    <div id="contenedor">  
        <div id="cabecera">Esta es la cabecera</div>  
        <div id="barra-lateral">Esta es la barra lateral</div>  
        <div id="contenido">Este es el contenido principal</div>  
        <div id="pie">Este es el pie de página</div>  
    </div>  
</body>  
</html>
```

Si queremos agregar un espacio entre las palabras y los bordes solo agregamos "`padding:5px;`" en todas las capas CSS.

## 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. Las siguientes imágenes muestran el aspecto de una página centrada a medida que aumenta la anchura de la ventana del navegador.



Figura 12.1. Página de anchura fija centrada mediante CSS



Figura 12.2. Página de anchura fija centrada mediante CSS



Figura 12.3. Página de anchura fija centrada mediante CSS

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 `<div>` y asignarle a ese `<div>` unos márgenes laterales automáticos. El `<div>` 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>
```

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

Las siguientes imágenes muestran cómo se adapta el diseño dinámico a la anchura de la ventana del navegador, mostrando cada vez más contenidos a medida que se agranda la ventana.



Figura 12.4. Página de anchura variable centrada mediante CSS



Figura 12.5. Página de anchura variable centrada mediante CSS



Figura 12.6. Página de anchura variable centrada mediante CSS

## Centrar una página verticalmente

Cuando se centra una página web de forma horizontal, sus márgenes laterales se adaptan dinámicamente de forma que la página siempre aparece en el centro de la ventana del navegador, independientemente de la anchura de la ventana. De la misma forma, cuando se centra una página web verticalmente, el objetivo es que sus contenidos aparezcan en el centro de la ventana del navegador y por tanto, que sus márgenes verticales se adapten de forma dinámica en función del tamaño de la ventana del navegador.

Aunque centrar una página web horizontalmente es muy sencillo, centrarla verticalmente es mucho más complicado. Afortunadamente, no es muy común que una página web aparezca centrada de forma vertical. El motivo es que la mayoría de páginas web son más altas que la ventana del navegador, por lo que no es posible centrarlas verticalmente.

A continuación se muestra la forma de centrar una página web respecto de la ventana del navegador, es decir, centrarla tanto horizontalmente como verticalmente.

Siguiendo el mismo razonamiento que el planteado para centrar la página horizontalmente, se podrían utilizar las siguientes reglas CSS para centrar la página respecto de la ventana del navegador:

```
#contenedor {  
width: 300px;  
height: 250px;  
margin: auto;  
}  
  
<body>  
<div id="contenedor">  
<h1>Lorem ipsum dolor sit amet</h1>  
...  
</div>  
</body>
```

Si el valor `auto` se puede utilizar para que los márgenes laterales se adapten dinámicamente, también debería ser posible utilizar el valor `auto` para los márgenes verticales. Desafortunadamente, la propiedad `margin: auto` no funciona tal y como se espera para los márgenes verticales y la página no se muestra centrada.

La solución correcta para centrar verticalmente una página web se basa en el **posicionamiento absoluto** e implica realizar un cálculo matemático sencillo. A continuación se muestra el esquema gráfico de los cuatro pasos necesarios para centrar una página web en la ventana del navegador:

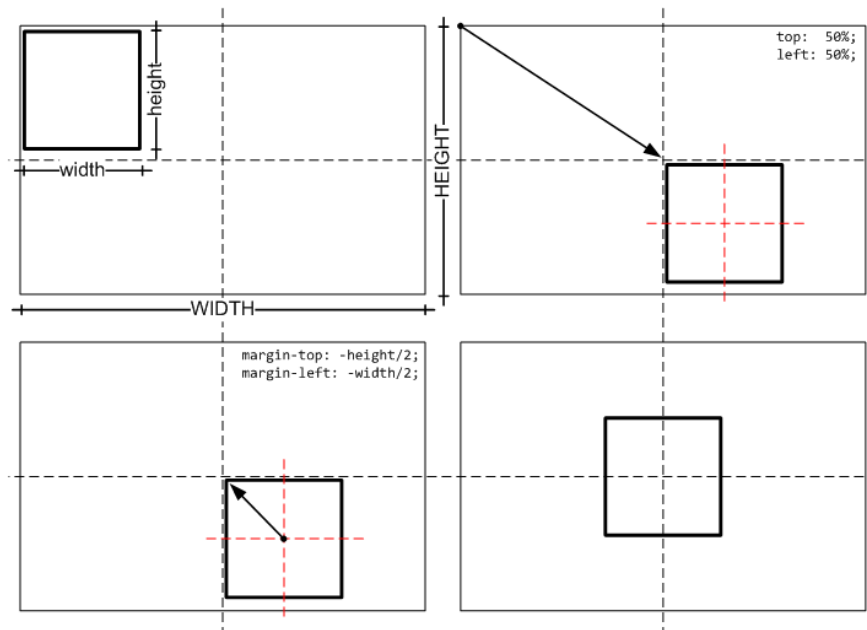


Figura 12.7. Pasos necesarios para centrar verticalmente una página web

En primer lugar, se asigna una altura y una anchura al elemento que encierra todos los contenidos de la página. En la primera figura, los contenidos de la página tienen una anchura llamada `width` y una altura llamada `height` que son menores que la anchura y altura de la ventana del navegador. En el siguiente ejemplo, se supone que tanto la anchura como la altura de la página es igual a 500px:

```
#contenedor {
width: 500px;
height: 500px;
}

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

A continuación, se posiciona de forma absoluta el elemento `contenedor` y se asigna un valor de 50% tanto a la propiedad `top` como a la propiedad `left`. El resultado es que la esquina superior izquierda del elemento `contenedor` se posiciona en el centro de la ventana del navegador:

```
#contenedor {
width: 500px;
height: 500px;

position: absolute;
top: 50%;
left: 50%;
}
```

Si la página se debe mostrar en el centro de la ventana del navegador, es necesario desplazar hacia **arriba y hacia la izquierda** los contenidos de la página web. Para determinar el desplazamiento necesario, se realiza un cálculo matemático sencillo. Como se ve en la tercera figura del esquema anterior, el punto central de la página debe desplazarse hasta el centro de la ventana del navegador.

Como se desprende de la imagen anterior, la página web debe moverse hacia arriba una cantidad igual a la mitad de su altura y debe desplazarse hacia la izquierda una cantidad equivalente a la mitad de su anchura. Utilizando las propiedades `margin-top` y `margin-left` con valores negativos, la página se desplaza hasta el centro de la ventana del navegador.

```
#contenedor {  
width: 500px;  
height: 500px;  
  
position: absolute;  
top: 50%;  
left: 50%;  
  
margin-top: -250px;    /* height/2 = 500px / 2 */  
margin-left: -250px;   /* width/2 = 500px / 2 */  
}
```

Con las reglas CSS anteriores, la página web siempre aparece centrada verticalmente y horizontalmente respecto de la ventana del navegador. El motivo es que la anchura/altura de la página son fijas (propiedades `width` y `height`), el desplazamiento necesario para centrarla también es fijo (propiedades `margin-top` y `margin-left`) y el desplazamiento hasta el centro de la ventana del navegador se calcula dinámicamente gracias al uso de porcentajes en las propiedades `top` y `left`.

Para centrar una página sólo verticalmente, se debe prescindir tanto del posicionamiento horizontal como del desplazamiento horizontal:

```
#contenedor {  
width: 500px;  
height: 500px;  
  
position: absolute;  
top: 50%;  
  
margin-top: -250px;    /* height/2 = 500px / 2 */  
}
```



## Diseño a 2 columnas con cabecera y pie de página

El objetivo de este diseño es definir una estructura de página con cabecera y pie, un menú lateral de navegación y una zona de contenidos. La anchura de la página se fija en 700px, la anchura del menú es de 150px y la anchura de los contenidos es de 550px:

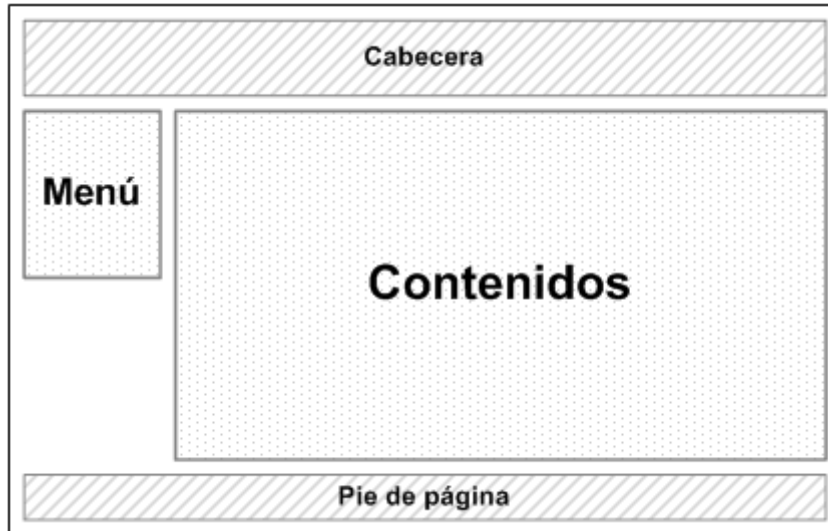


Figura 12.8. Esquema del diseño a 2 columnas con cabecera y pie de página

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`.

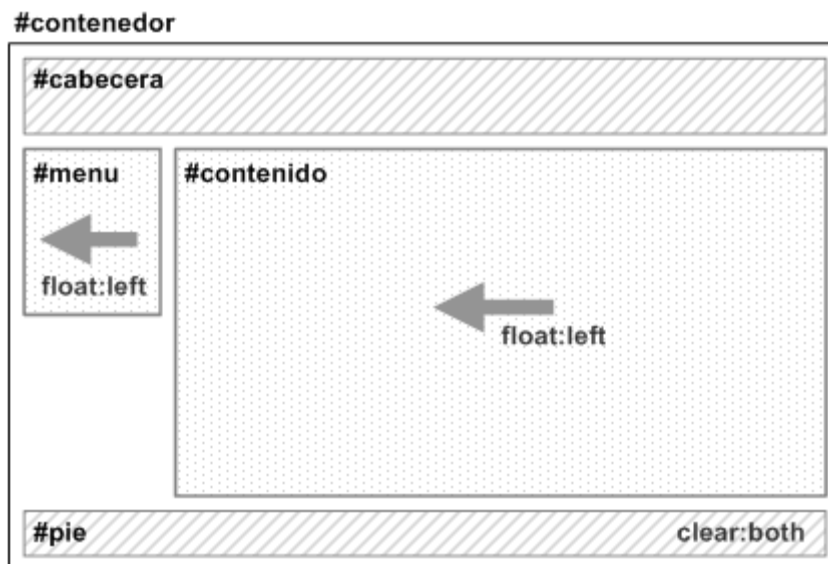


Figura 12.9. Propiedades CSS necesarias en el diseño a dos columnas con cabecera y pie de página

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

```
#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. Para crear diseños de anchura fija, basta con establecer las anchuras de los bloques en píxel.

### 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:

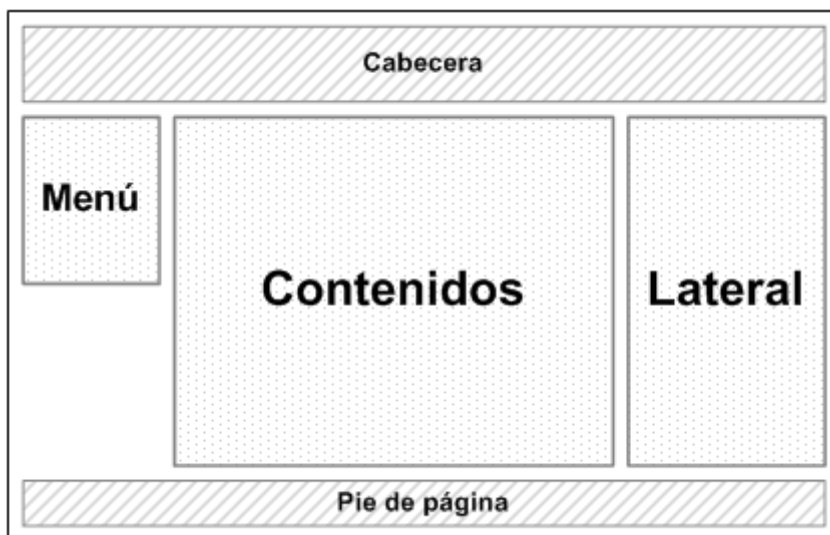


Figura 12.10. Esquema del diseño a tres columnas con cabecera y pie de página

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

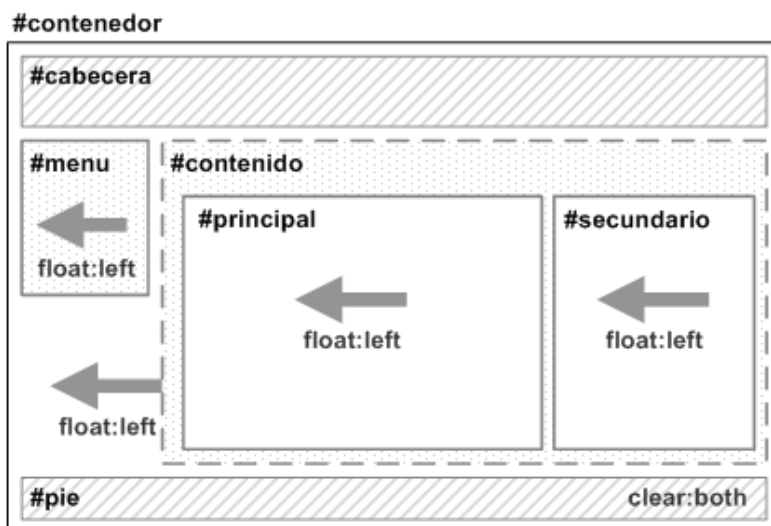


Figura 12.11. Propiedades CSS necesarias en el diseño a 3 columnas con cabecera y pie de página

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. Para definir una página con anchura fija, solamente es necesario sustituir las anchuras en porcentajes por anchuras en píxel.

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

## Posicionamiento y visualización

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

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

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

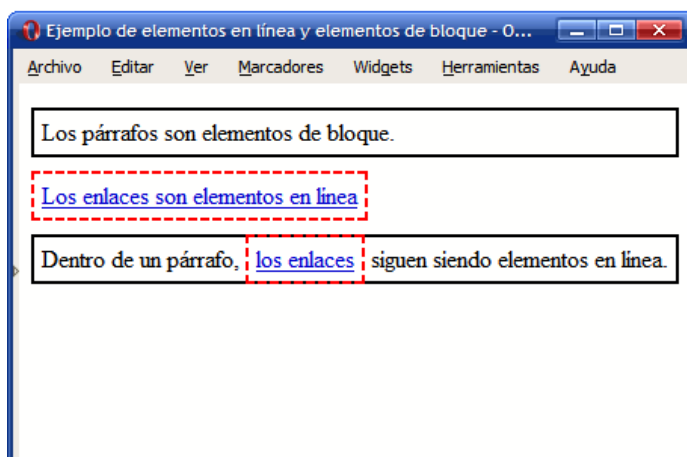
En este capítulo se muestran los cinco tipos de posicionamientos definidos para las cajas y se presentan otras propiedades que afectan a la forma en la que se visualizan las cajas.

### 1. Tipos de elementos

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

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

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



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

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

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

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

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

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

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

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

## 2. Posicionamiento

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

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

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

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

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

<b>position</b>	Posicionamiento
<b>Valores</b>	<code>static</code>   <code>relative</code>   <code>absolute</code>   <code>fixed</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	<code>static</code>
<b>Descripción</b>	Selecciona el posicionamiento con el que se mostrará el elemento

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

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

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

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

<b>Top right bottom left</b>	Desplazamiento superior Desplazamiento lateral derecho Desplazamiento inferior Desplazamiento lateral izquierdo
<b>Valores</b>	<medida>   <porcentaje>   auto   inherit
<b>Se aplica a</b>	Todos los elementos posicionados
<b>Valor inicial</b>	auto
<b>Descripción</b>	Indican el desplazamiento horizontal y vertical del elemento respecto de su posición original

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

En cualquiera de los dos casos, si el desplazamiento se indica en forma de porcentaje, se refiere al porcentaje sobre la anchura (propiedades `right` y `left`) o altura (propiedades `top` y `bottom`) del elemento.

### 3. Posicionamiento normal

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, ninguna caja se desplaza respecto de su posición original, por lo que sólo se tiene en cuenta si el elemento es de bloque o en línea.

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

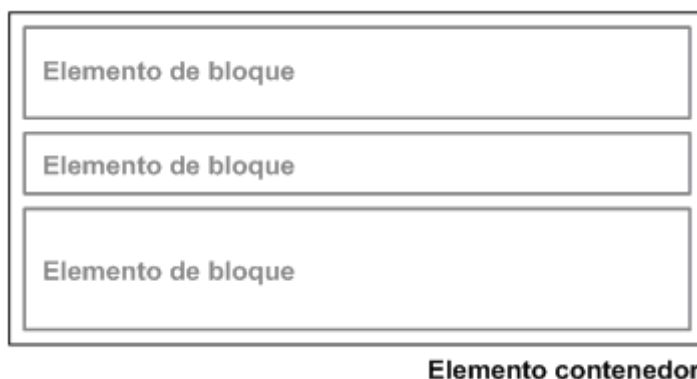


Figura 2. Posicionamiento normal de los elementos de bloque



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

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

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

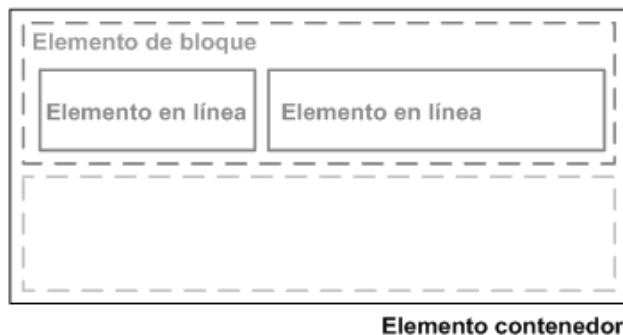


Figura 5.3. Posicionamiento normal de los elementos en línea

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

## 4. Posicionamiento relativo

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

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

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

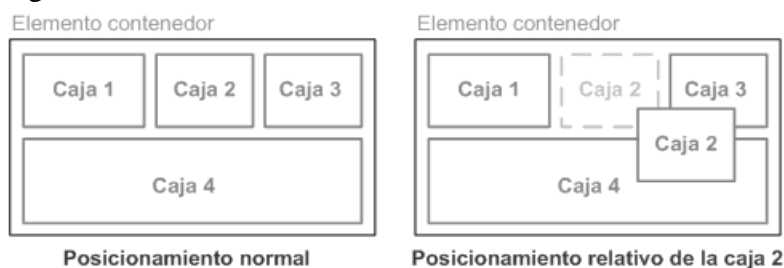


Figura 5.4. Ejemplo de posicionamiento relativo de un elemento

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

La propiedad `left` desplaza la caja hacia su derecha, la propiedad `right` la desplaza hacia su izquierda, la posición `top` desplaza la caja de forma descendente y la propiedad `bottom` desplaza la caja de forma ascendente. Si se utilizan valores negativos en estas propiedades, su efecto es justamente el inverso.

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

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

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

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

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

El siguiente ejemplo muestra tres imágenes posicionadas de forma normal:

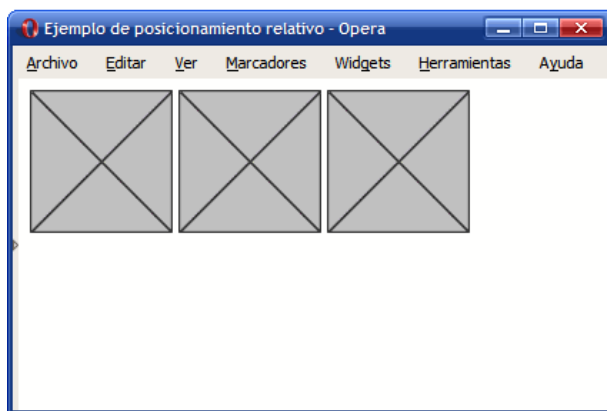


Figura 5.5. Elementos posicionados de forma normal

Aplicando el posicionamiento relativo, se desplaza la primera imagen de forma descendente:

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

```
  
  

```

El aspecto que muestran ahora las imágenes es el siguiente:

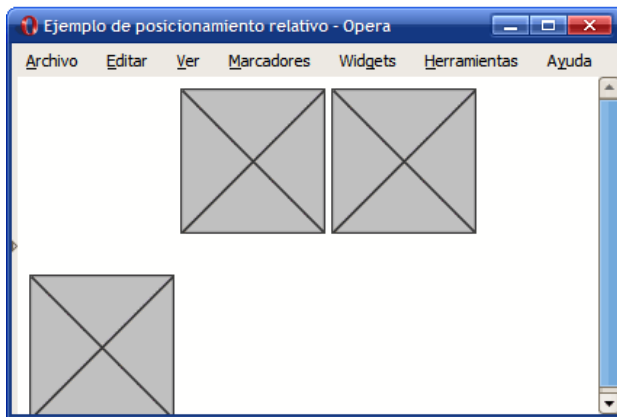


Figura 5.6. Elemento posicionado de forma relativa

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

## 5. Posicionamiento absoluto

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

Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página la ignoran y ocupan el lugar original ocupado por la caja posicionada. Al igual que en el posicionamiento relativo, cuando se posiciona de forma absoluta una caja es probable que se produzcan solapamientos con otras cajas.

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



Figura 5.7. Ejemplo de posicionamiento absoluto de un elemento

La caja 2 está posicionada de forma absoluta, lo que implica que el resto de elementos ignoran que esa caja exista. Por este motivo, la caja 3 deja su lugar original y pasa a ocupar el hueco dejado por la caja 2.

En el estándar de CSS, esta característica de las cajas posicionadas de forma absoluta se explica como que **la caja sale por completo del flujo normal del documento**. De hecho, las cajas posicionadas de forma absoluta parece que están en un nivel diferente al resto de elementos de la página.

Por otra parte, el desplazamiento de una caja posicionada de forma absoluta se indica mediante las propiedades `top`, `right`, `bottom` y `left`. A diferencia de posicionamiento relativo, en este caso la referencia de los valores de esas propiedades es el origen de coordenadas **de su primer elemento contenedor posicionado**.

Determinar el origen de coordenadas a partir del cual se desplaza una caja posicionada de forma absoluta es un proceso complejo que se compone de los siguientes pasos:

- Se buscan todos los elementos contenedores de la caja hasta llegar al elemento `<body>` de la página.
- Se recorren todos los elementos contenedores empezando por el más cercano a la caja y llegando hasta el `<body>`
- De todos ellos, el navegador se queda con el primer elemento contenedor que esté posicionado de cualquier forma diferente a `position: static`
- La esquina superior izquierda de ese elemento contenedor posicionado es el **origen de coordenadas**.

Una vez obtenido el origen de coordenadas, se interpretan los valores de las propiedades `top`, `right`, `bottom` y `left` respecto a ese origen y se desplaza la caja hasta su nueva posición.

En los siguientes ejemplos, se va a utilizar la página HTML que muestra la siguiente imagen:

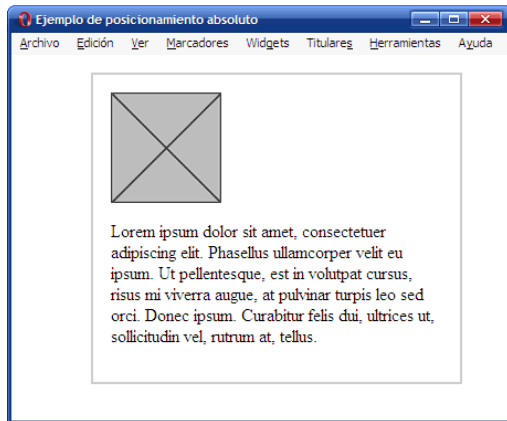


Figura 5.8. Situación original antes de modificar el posicionamiento

A continuación, se muestra el código HTML y CSS de la página original:

```
div {  
  border: 2px solid #CCC;  
  padding: 1em;  
  margin: 1em 0 1em 4em;  
  width: 300px;  
}  
  
<div>  
    
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus  
    ullamcorper veliteuipsum. Utpellentesque, est in volutpat cursus,  
    risus miviverra augue, at pulvinar turpis leo sed  
    orci. Donec ipsum. Curabitur felis dui, ultrices ut,  
    sollicitudin vel, rutrum at, tellus.</p>  
</div>
```

En primer lugar, se posiciona de forma absoluta la imagen mediante la propiedad `position` y se indica su nueva posición mediante las propiedades `top` y `left`:

```
Div img {  
  position: absolute;  
  top: 50px;  
  left: 50px;  
}
```

El resultado visual se muestra en la siguiente imagen:

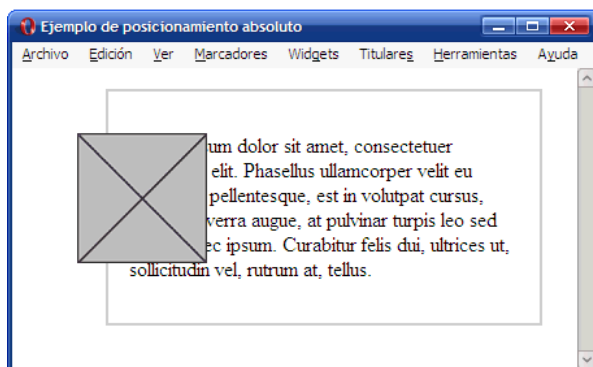


Figura 5.9. Imagen posicionada de forma absoluta

La imagen posicionada de forma absoluta no toma como origen de coordenadas la esquina superior izquierda de su elemento contenedor `<div>`, sino que su referencia es la esquina superior izquierda de la página:

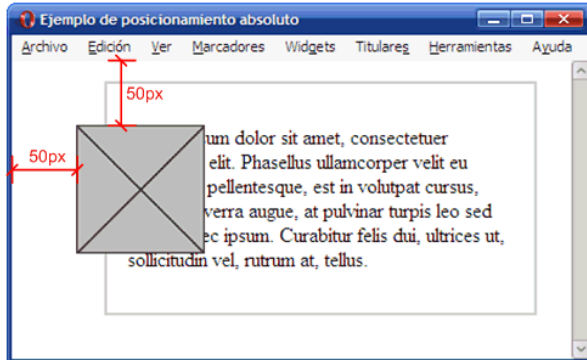


Figura 5.10. La referencia del posicionamiento absoluto es la página entera

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

1. Obtiene la lista de elementos contenedores de la imagen: `<div>` y `<body>`.
2. Recorre la lista de elementos desde el más cercano a la imagen (el `<div>`) hasta terminar en el `<body>` buscando el primer elemento contenedor que esté posicionado.
3. El primer elemento contenedor es el `<div>`, pero su posicionamiento es el normal o estático, ya que ni siquiera tiene establecida la propiedad `position`.
4. Como el siguiente elemento contenedor es el `<body>`, el navegador establece directamente el origen de coordenadas en la esquina superior izquierda de la página.
5. A partir de ese origen de coordenadas, la caja se desplaza 50px hacia la derecha (`left: 50px`) y otros 50px de forma descendente (`top: 50px`).

Además, el párrafo que se mostraba debajo de la imagen sube y ocupa el lugar dejado por la imagen. El resultado es que el elemento `<div>` ahora sólo contiene el párrafo y la imagen se muestra en un nivel superior y cubre parcialmente los contenidos del párrafo.

A continuación, se posiciona de forma relativa el elemento `<div>` que contiene la imagen y el resto de reglas CSS no se modifican. La única propiedad añadida al `<div>` es `position: relative` por lo que el elemento contenedor se posiciona pero no se desplaza respecto de su posición original:

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

```
Div img {
  position: absolute;
```

```
top: 50px;  
left: 50px;  
}
```

La siguiente imagen muestra el resultado obtenido:

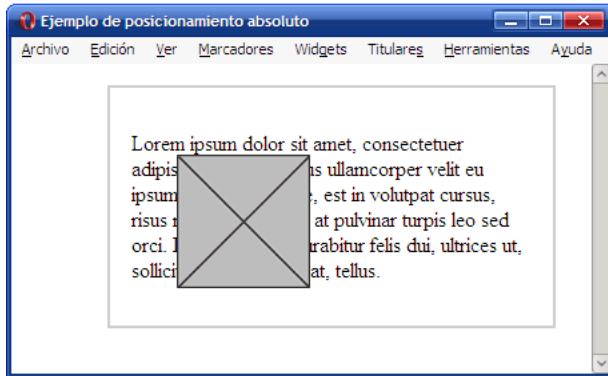


Figura 5.11. Imagen posicionada de forma absoluta

En este caso, el origen de coordenadas para determinar la nueva posición de la imagen corresponde a la esquina superior izquierda del elemento `<div>`:

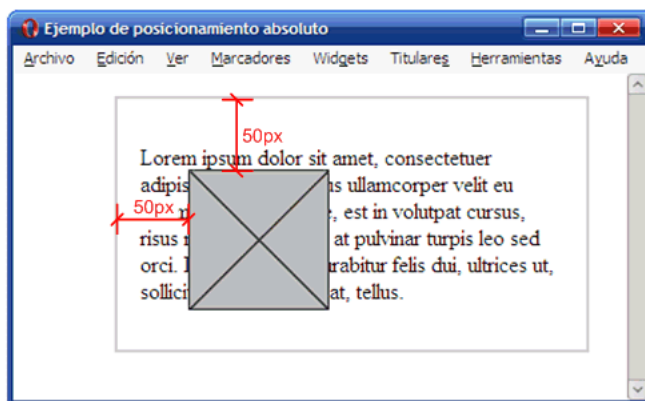


Figura 5.12. La referencia del posicionamiento absoluto es el elemento contenedor de la imagen

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

## 6. Posicionamiento fijo

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

Cuando una caja se posiciona de forma fija, la forma de obtener el origen de coordenadas para interpretar su desplazamiento es **idéntica al posicionamiento absoluto**.

De hecho, si el usuario no mueve la página HTML en la ventana del navegador, no existe ninguna diferencia entre estos dos modelos de posicionamiento.

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

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

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

## 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:

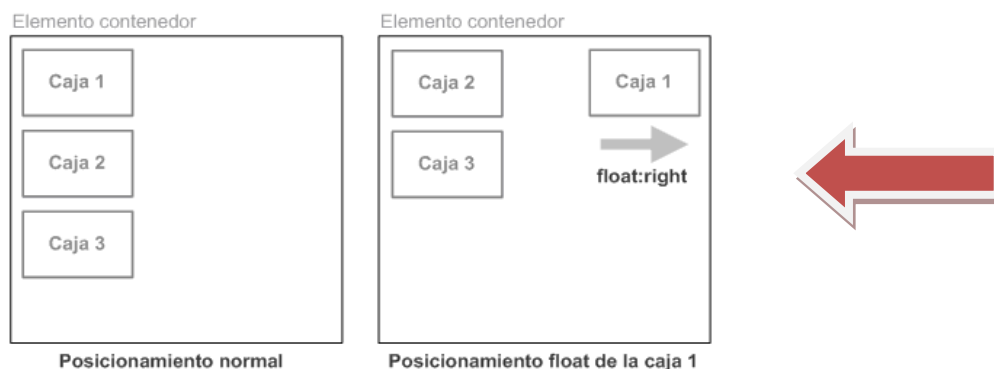


Figura 5.13. Ejemplo de posicionamiento float de una caja

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:

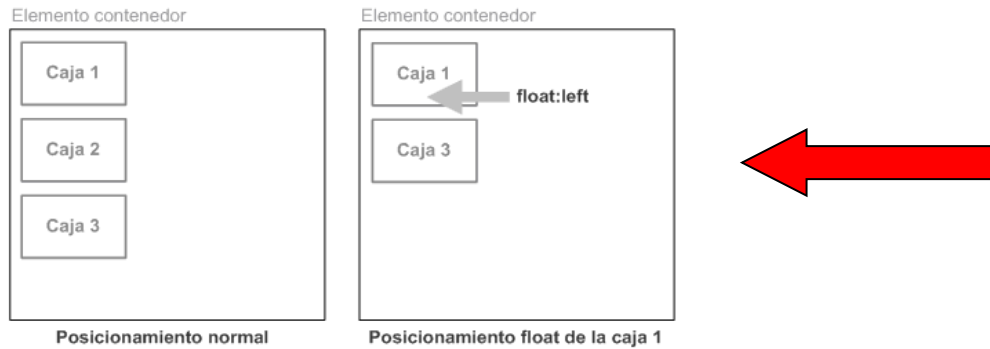


Figura 5.14. Ejemplo de posicionamiento float de una caja

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:

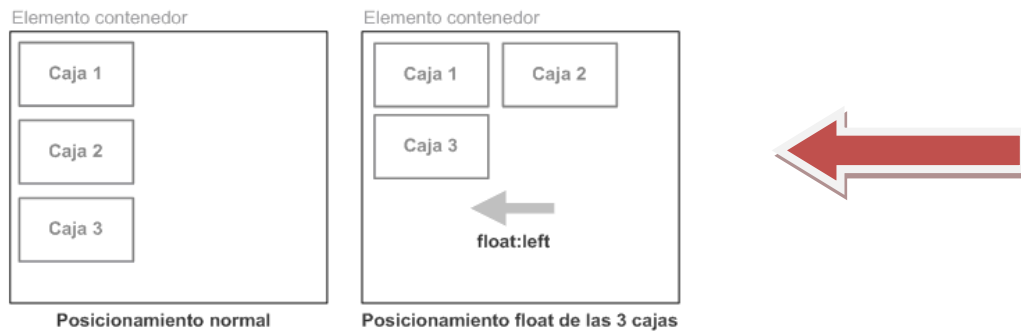


Figura 5.16. Ejemplo de posicionamiento float cuando no existe sitio suficiente

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`:

<b>float</b>	Posicionamiento float
<b>Valores</b>	left   right   none   inherit
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	none
<b>Descripción</b>	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:

A partir del código HTML proporcionado:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio posicionamiento float</title>
<style type="text/css">
```

```

</style>
</head>

<body>
<div>
&laquo; Anterior &nbsp; &nbsp; Siguiente &raquo;
</div>
</body>
</html>

```

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



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

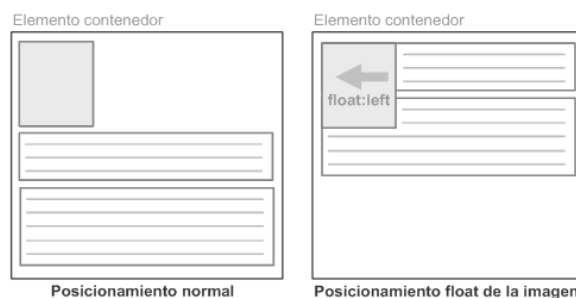


Figura 5.17. Elementos que fluyen alrededor de un elemento posicionado mediante float

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:

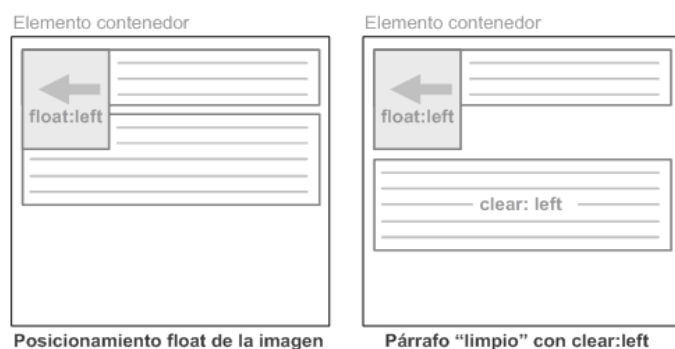


Figura 5.18. Forzando a que un elemento no fluya alrededor de otro elemento posicionado mediante float

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:

<b>clear</b>	Despejar los elementos flotantes adyacentes
<b>Valores</b>	none   left   right   both   inherit
<b>Se aplica a</b>	Todos los elementos de bloque
<b>Valor inicial</b>	none
<b>Descripción</b>	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.

Se considera el siguiente estilo

```
#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>
```

El resultado que se muestra es el siguiente

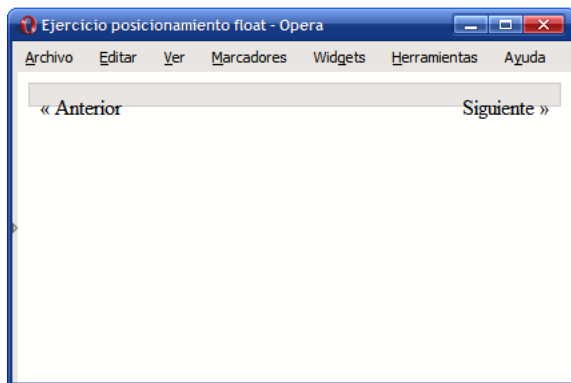


Figura 5.19. Visualización incorrecta de dos elementos posicionados mediante float

Como los dos elementos `<span>` creados dentro del elemento `<div>` se han posicionado mediante `float`, los dos han salido del flujo normal del documento. Así, el elemento `<div>` no tiene contenidos y por eso no llega a cubrir el texto de los dos elementos `<span>`:

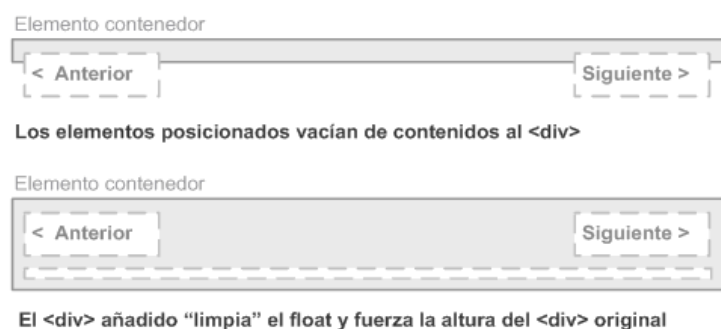


Figura 5.20. Esquema del problema y solución de la visualización incorrecta de dos elementos posicionados mediante float

La solución consiste en añadir un elemento adicional invisible que *limpie* el float forzando a que el <div> original cubra completamente los dos elementos <span>. El código HTML y CSS final se muestra a continuación:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio posicionamiento float</title>
<style type="text/css">
div#paginacion {
border: 1px solid #CCC;
background-color: #E0E0E0;
padding: .5em;
}

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

div.clear { clear: both; }
</style>
</head>

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

Al añadir un <div> con la propiedad `clear: both`, se tiene la seguridad de que el <div> añadido se va a mostrar debajo de cualquier elemento posicionado con float y por tanto, se asegura que el <div> original tenga la altura necesaria como para encerrar a todos sus contenidos posicionados con float.

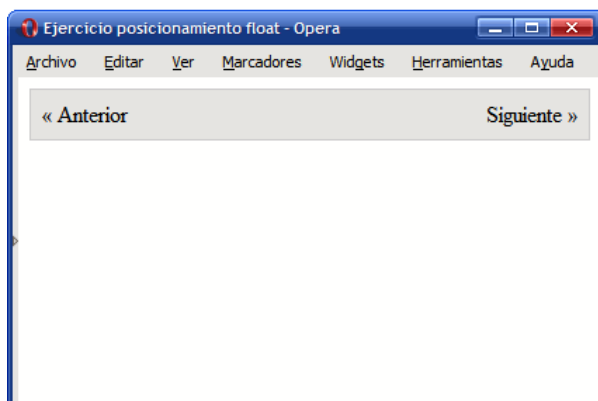


Figura 5.21. Visualización correcta de dos elementos posicionados mediante float

Además de un elemento <div> invisible, también se puede utilizar un <p> invisible o un <hr/> invisible.

### Otra solución, mucho más rápida es la siguiente:

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

## 5.8. Visualización

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

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

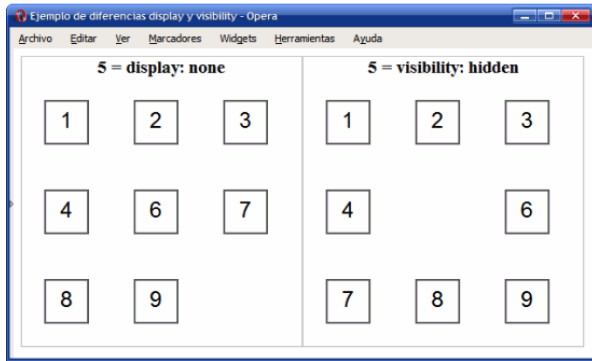
### 5.8.1. Propiedades `display` y `visibility`

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

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

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

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



#### Diferencias visuales entre las propiedades `display` y `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`:

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

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

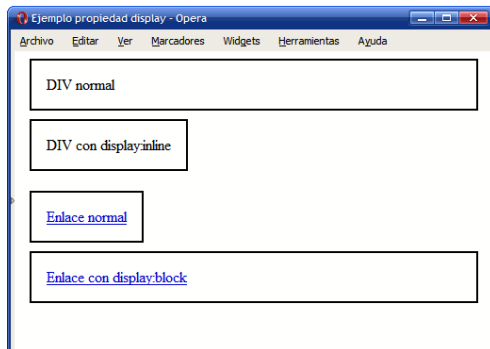
Los valores más utilizados son `inline`, `block` y `none`. El valor `block` muestra un elemento como si fuera un elemento de bloque, independientemente del tipo de



elemento que se trate. El valor `inline` visualiza un elemento en forma de elemento en línea, independientemente del tipo de elemento que se trate.

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

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



### Ejemplo de propiedad display

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>
```

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

Por su parte, la definición completa de la propiedad `visibility` es mucho más sencilla:

<b>visibility</b>	Visibilidad de un elemento
<b>Valores</b>	visible   hidden   collapse   inherit
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	visible
<b>Descripción</b>	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 `col`

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

### 5.8.2. Relación entre `display`, `float` y `position`

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

- 
1. Si `display` vale `none`, se ignoran las propiedades `float` y `position` y la caja no se muestra en la página.
  2. 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`.
  3. En cualquier otro caso, si `float` tiene un valor distinto de `none`, la caja se posiciona de forma flotante y la propiedad `display` vale `block` tanto para los elementos en línea como para los elementos de bloque.

### 5.8.3. Propiedad `overflow`

Normalmente, los contenidos de un elemento se pueden mostrar en el espacio reservado para ese elemento. Sin embargo, en algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda.

La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad `width` y/o `height`. Otra situación habitual es la de las líneas muy largas contenidas dentro de un elemento `<pre>`, que hacen que la página entera sea demasiado ancha.

CSS define la propiedad `overflow` para controlar la forma en la que se visualizan los contenidos que sobresalen de sus elementos.

<b>overflow</b>	Parte sobrante de un elemento
<b>Valores</b>	visible   hidden   scroll   auto   inherit
<b>Se aplica a</b>	Elementos de bloque y celdas de tablas
<b>Valor inicial</b>	visible
<b>Descripción</b>	Permite controlar los contenidos sobrantes de un elemento

Los valores de la propiedad `overflow` tienen el siguiente significado:

- `visible`: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.
- `hidden`: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.
- `scroll`: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de *scroll* que permiten visualizar el resto del contenido.
- `auto`: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad `scroll`.

La siguiente imagen muestra un ejemplo de los tres valores típicos de la propiedad `overflow`:



Ejemplo de propiedad `overflow`

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

```
div{
  display: inline;
  float: left;
  margin: 1em;
  padding: .3em;
  border: 2px solid #555;
  width: 100px;
  height: 150px;
```

```
font: 1em Arial, Helvetica, sans-serif;
}
```

```
<div><h1>overflow: visible</h1>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Cras vitae dolor eu enim dignissim lacinia. Maecenas
blandit. Morbi mi.</div>
```

```
<div style="overflow: hidden"><h1>overflow: hidden</h1>Lorem ipsum dolor
sit amet, consectetur adipiscing elit. Cras vitae dolor eu enim dignissim
lacinia. Maecenas blandit. Morbi mi.</div>
```

```
<div style="overflow: scroll"><h1>overflow: scroll</h1>Lorem ipsum dolor
sit
amet, consectetur adipiscing elit. Cras vitae dolor eu enim dignissim
lacinia.
Maecenas blandit. Morbi mi.</div>
```

#### 5.8.4. Propiedad z-index

Además de posicionar una caja de forma horizontal y vertical, CSS permite controlar la posición tridimensional de las cajas posicionadas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional de un elemento se establece sobre un tercer eje llamado Z y se controla mediante la propiedad `z-index`. Utilizando esta propiedad es posible crear páginas complejas con varios niveles o capas.

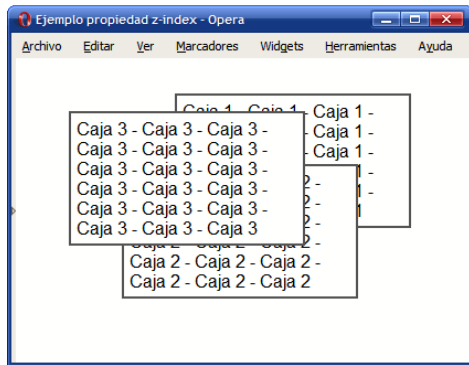
A continuación se muestra la definición formal de la propiedad `z-index`:

<b>z-index</b>	Orden tridimensional
<b>Valores</b>	auto   <numero>   inherit
<b>Se aplica a</b>	Elementos que han sido posicionados explícitamente
<b>Valor inicial</b>	auto
<b>Descripción</b>	Establece el nivel tridimensional en el que se muestra el elemento

El valor más común de la propiedad `z-index` es un número entero. Aunque la especificación oficial permite los números negativos, en general se considera el número 0 como el nivel más bajo.

Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja. Un elemento con `z-index: 10` se muestra por encima de los elementos con `z-index: 8` o `z-index: 9`, pero por debajo de elementos con `z-index: 20` o `z-index: 50`.

La siguiente imagen muestra un ejemplo de uso de la propiedad `z-index`:



### Ejemplo de propiedad z-index

El código HTML y CSS del ejemplo anterior es el siguiente:

```
div{position: absolute; }
#cajal{z-index: 5; top: 1em; left: 8em;}
#caja2{z-index: 15; top: 5em; left: 5em;}
#caja3{z-index: 25; top: 2em; left: 2em;}
```

```
<div id="cajal">Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 -  
Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 -  
Caja 1 - Caja 1 - Caja 1 - Caja 1</div>
```

```
<div id="caja2">Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 -  
Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 -  
Caja 2 - Caja 2 - Caja 2 - Caja 2</div>
```

```
<div id="caja3">Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 -  
Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3  
-  
Caja 3 - Caja 3 - Caja 3 - Caja 3</div>
```

La propiedad `z-index` sólo tiene efecto en los elementos posicionados, por lo que es obligatorio que la propiedad `z-index` vaya acompañada de la propiedad `position`. Si debes posicionar un elemento pero no quieres moverlo de su posición original ni afectar al resto de elementos de la página, puedes utilizar el posicionamiento relativo (`position: relative`).