# Posicionamiento con CSS en 10 Ejemplos

## Código General HTML

```
<div id="example">
	<div id="div-0">
		<p>id = div-0</p>
	</div>

	<div id="div-1">

		<div id="div-1-padding">
			<p>id = div-1</p>
			<div id="div-1a">
				<p>id = div-1a</p>
				<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam. Phasellus ultrices viverra velit.</p>
			</div>
			<div id="div-1b">
				<p>id = div-1b</p>
				<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam. Phasellus ultrices viverra velit. Nam mattis, arcu ut bibendum commodo, magna nisi tincidunt tortor, quis accumsan augue ipsum id lorem.</p>
			</div>
			<div id="div-1c">
				<p>id = div-1c</p>
			</div>
		</div>
	</div>

	<div id="div-2">
		<p>id = div-2</p>
	</div>
</div>
```

## Código General CSS

```
#div-0, #div-2 {
background-color:#88d;
color:#000;
}
#div-1 {
width:400px;
background-color:#000;
color:#fff;
}
#div-1-padding {
```

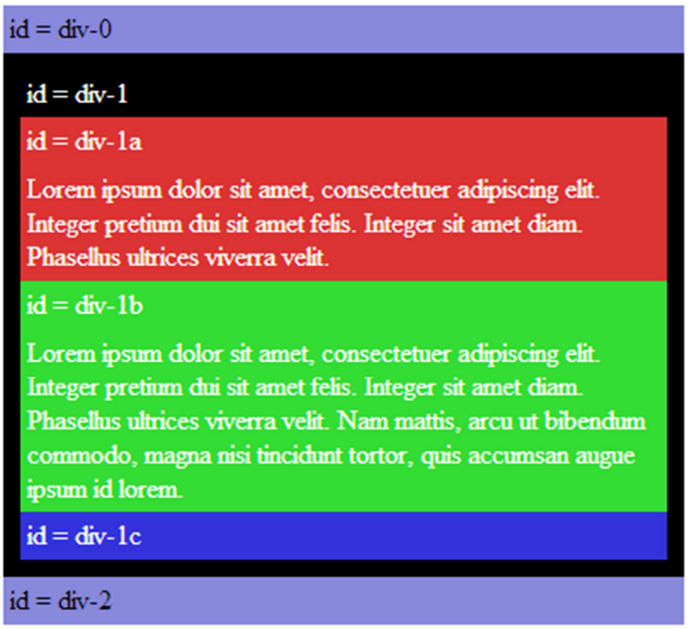
```
padding:10px;
}
#div-1a {
background-color:#d33;
color:#fff;
}
#div-1b {
background-color:#3d3;
color:#fff;
}
#div-1c {
background-color:#33d;
color:#fff;
}
```

## Ejemplo 1: position:static

El valor predeterminado para todos los elementos es *position:static*, que posiciona al elemento según el lugar donde fue definido en el HTML.

Por lo general no hay necesidad de definir *position:static*, a menos que queramos caerle encima a alguna regla anterior que haya modificado el posicionamiento.

```
#div-1 {
  position:static;
}
```



## Ejemplo 2: position:relative

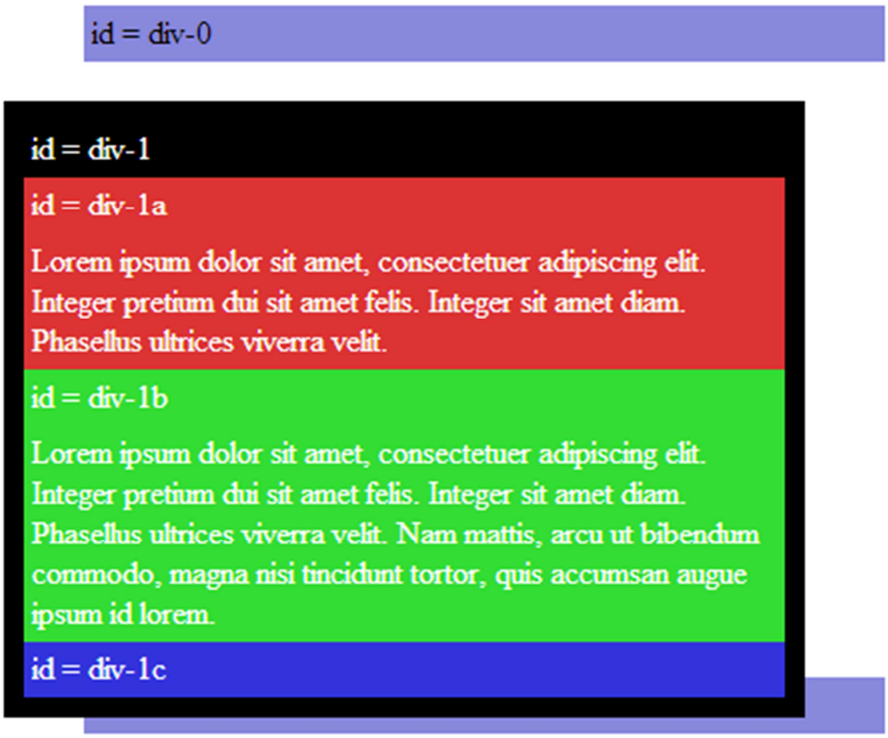
Si le definimos *position:relative* a un elemento, entonces podemos utilizar *top*,*bottom*, *left* y *right* para mover el elemento con relación a la posición en la que aparecería normalmente en el documento.

Movamos el div-1 20 pixeles hacia abajo y 40 pixeles hacia la izquierda:

```
#div-1 {
 position:relative;
 top:20px;
 left:-40px;
}
```

Fíjate que en el lugar donde hubiese aparecido el div-1 ahora hay un espacio en blanco. El elemento siguiente (div-2) no cambió de posición porque ese espacio en blanco le sigue perteneciendo al div-1, aunque lo hayamos movido.

Parecería que *position:relative* no es muy útil, pero tendrá una tarea muy importante más adelante.



## Ejemplo 3: position:absolute

Cuando utilizamos *position:absolute*, el elemento es removido del documento y colocado exactamente donde nosotros queremos.

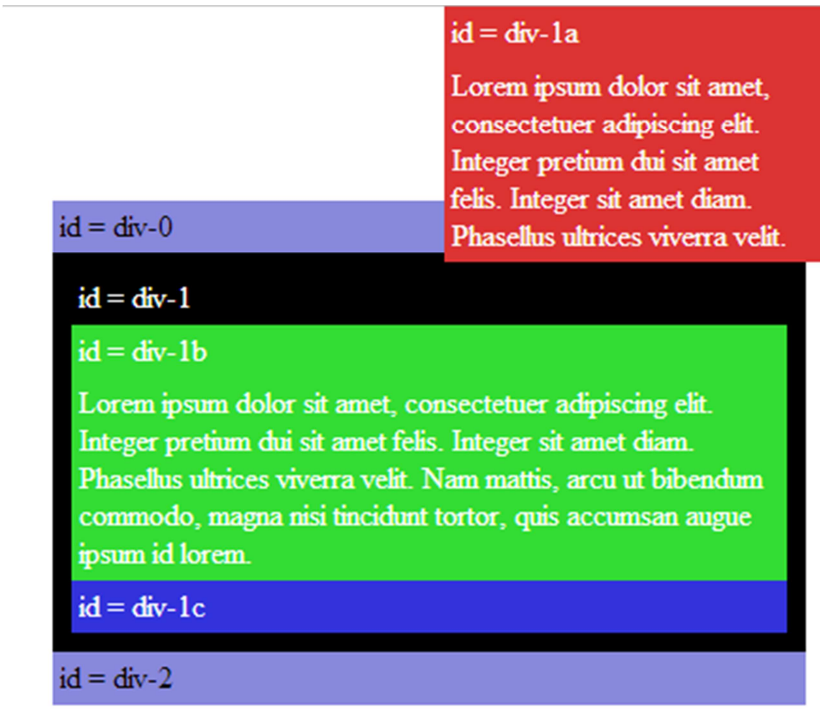
Movamos el div-1a a la esquina superior derecha de la página:

```
#div-1a {
 position:absolute;
 top:0;
 right:0;
 width:200px;
}
```

Fíjate que esta vez, como el div-1a fue removido del documento, los otros elementos se posicionaron de manera distinta: div-1b, div-1c y div-2 se movieron hacia arriba para ocupar el espacio que antes era de div-1a.

También fíjate que el div-1a fue colocado en la esquina superior derecha de la página. Es divertido poder colocar elementos directamente en la página, pero usualmente no es de gran utilidad.

Lo que realmente queremos hacer es colocar el div-1a en la esquina superior del div-1. Aquí es donde *position:relative* vuelve a tener importancia.



## Ejemplo 4: position:relative + position:absolute

Si le definimos *position:relative* al div-1, cualquier elemento que le coloquemos dentro se posicionará con relación a div-1.

Si luego le definimos *position:absolute* al div-1a, podemos colocarlo en la esquina superior derecha del div-1:

```
#div-1 {
 position:relative;
}
#div-1a {
 position:absolute;
 top:0;
 right:0;
 width:200px;
}
```

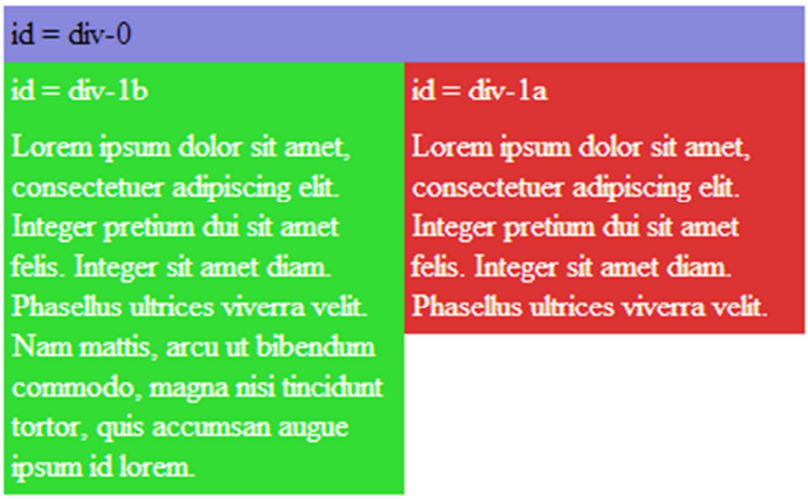
## Ejemplo 5: dos columnas absolutas

¡Ahora podemos crear un layout de dos columnas utilizando posicionamiento relativo y absoluto!

```
#div-1 {
 position:relative;
}
#div-1a {
 position:absolute;
 top:0;
 right:0;
 width:200px;
}
#div-1b {
 position:absolute;
 top:0;
 left:0;
 width:200px;
}
```

Una ventaja de utilizar posicionamiento absoluto es que podemos colocar los elementos en cualquier posición en la página, sin importar el orden en el que aparezcan en el HTML. Por ejemplo, decidimos posicionar div-1b antes que div-1a.

Un momento, ¿qué sucedió con los demás elementos? Están siendo ocultados por los elementos de posición absoluta. ¿Qué podemos hacer al respecto?

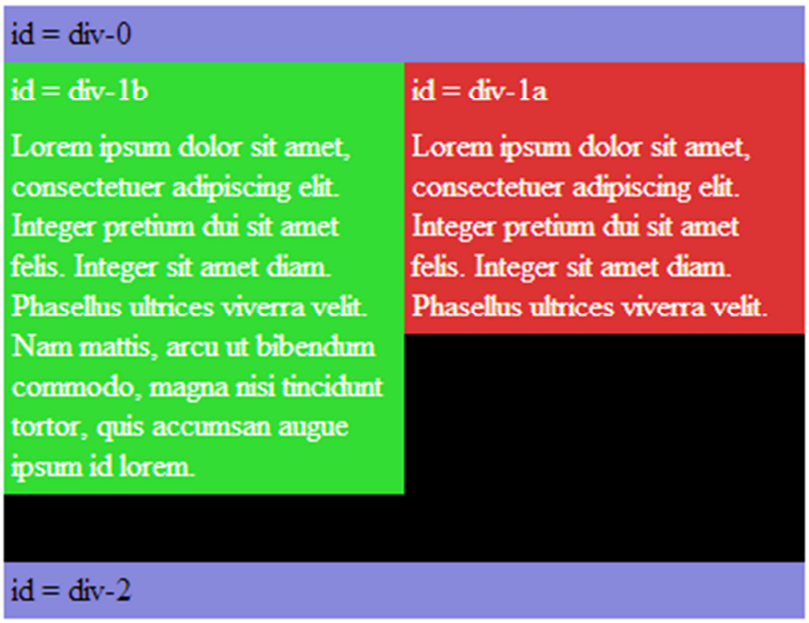


## Ejemplo 6: dos columnas altura absoluta

Una opción sería definir una altura fija para los elementos.

```
#div-1 {
 position:relative;
 height:250px;
}
#div-1a {
 position:absolute;
 top:0;
 right:0;
 width:200px;
}
#div-1b {
 position:absolute;
 top:0;
 left:0;
 width:200px;
}
```

Sin embargo, esta no es una solución viable para la mayoría de los diseños, ya que por lo general no sabemos de antemano cuánto texto tendrá cada elemento, ni cuál será el tamaño de fuente a utilizar.

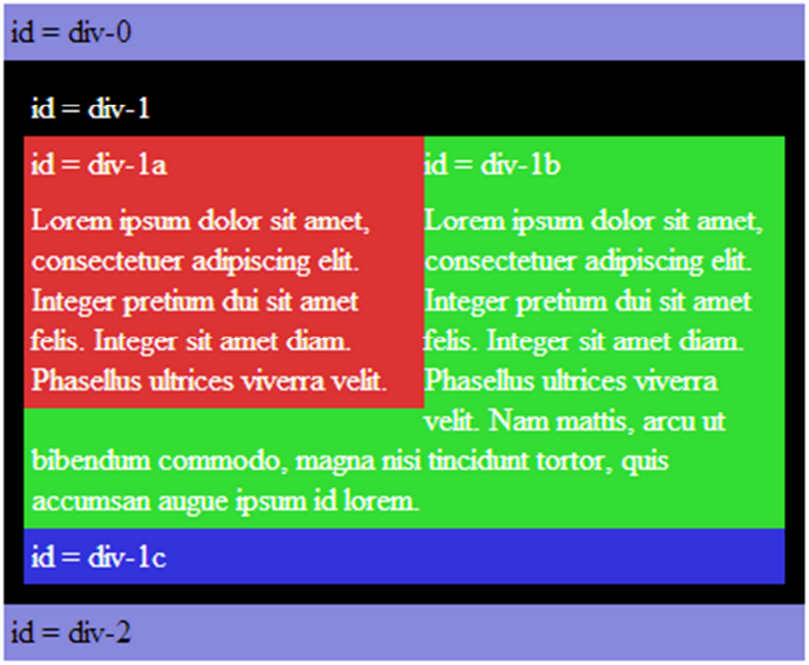


## Ejemplo 7: float

Para columnas de altura variable, el posicionamiento absoluto no funciona, asi que debemos buscar otra alternativa.

Podemos "flotar" un elemento para empujarlo lo más que se pueda hacia la derecha o izquierda, permitiendo así que el texto rodee al elemento. Esta técnica se utiliza normalmente con imágenes, pero también se puede utilizar para crear layouts complejos.

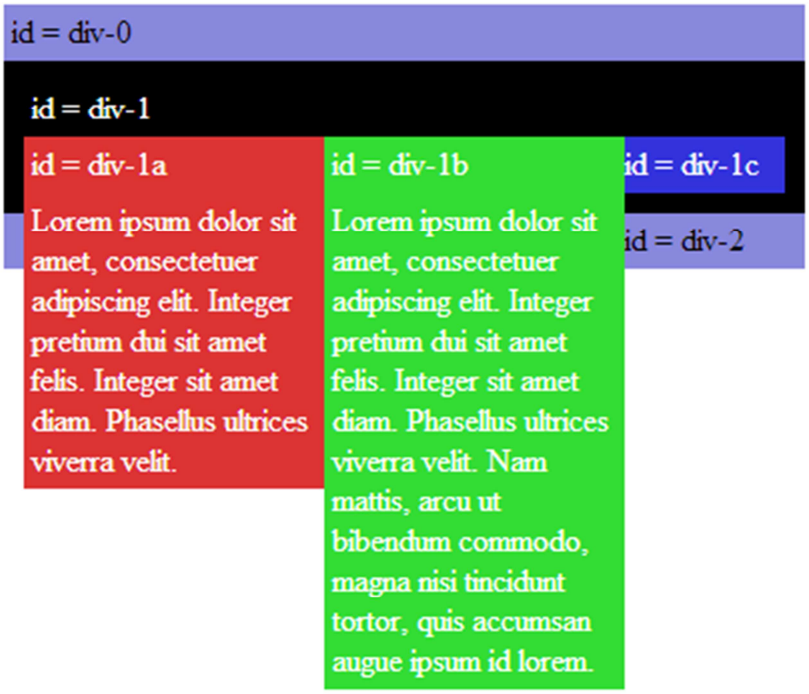
```
#div-1a {
 float:left;
 width:200px;
}
```



## Ejemplo 8: columnas flotantes

Si flotamos una columna hacia la izquierda y la otra hacia la derecha, entonces quedarán una al lado de la otra.

```
#div-1a {
 float:left;
 width:150px;
}
#div-1b {
 float:left;
 width:150px;
}
```



## Ejemplo 9: columnas flotantes con clear

Después de los elementos flotantes podemos añadir un "clear" para empujar el resto del contenido hacia abajo.

```
#div-1a {
 float:left;
 width:190px;
}
#div-1b {
 float:left;
 width:190px;
}
#div-1c {
 clear:both;
}
```


## Aclaraciones

Estos son ejemplos muy sencillos, que no deben generar errores o bugs.

Esta es una traducción a español del artículo:

[Learn CSS Positioning in Ten Steps]