

# Positioning

## CSS Positioning

To use CSS for layout effectively, it helps to know how it's used to position page content. This article gives an overview of the methods and rules that govern visual rendering in the CSS2 specification. It also points out some things to watch out for.

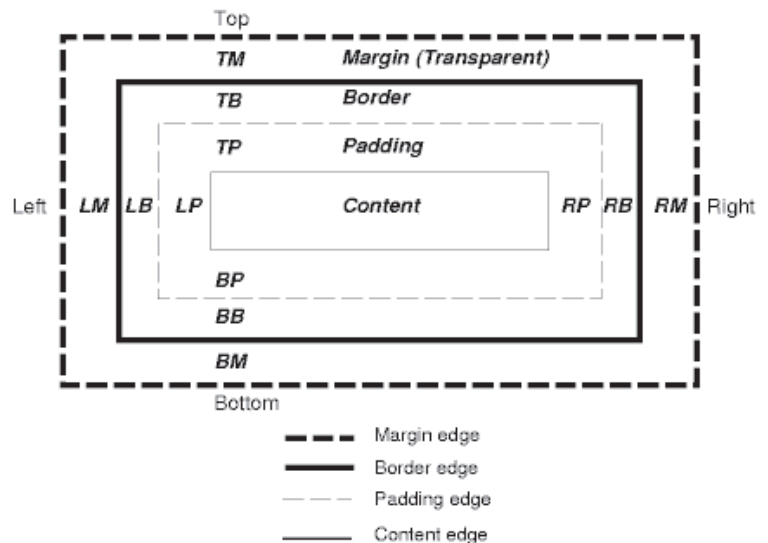
Although the specification applies to any device for displaying web pages, this article focuses on how it works in browsers. Many details are left out for the sake of simplicity. For a definitive reference, see the standards publication.

It's important to remember that a given browser may not support a given feature or may even implement it incorrectly. Also, there is some leeway provided within the standards where individual browsers are free to deal with situations as they please. Where appropriate these inconsistencies are noted.

## The Box Model

To understand positioning in CSS you must first understand the box model. For display purposes, every element in a document is considered to be a rectangular box which has a content area surrounded by padding, a border and margins. The illustration below shows these various parts.

Margins are always transparent. Borders come in various styles. Background settings for an element apply to the the area just inside the borders which includes both the padding and content areas. For purposes of illustration however, the padding area is shown in a slightly darker color.



When referring to boxes throughout this article, the term "margin edge," "border edge", etc. means the the outer boundary of the corresponding box area as shown above.

Margins, borders and padding are all optional but for purposes of calculating positions and sizes they are given a default width of zero if not specified. Different widths can be set for each individual side (top, right, bottom and left) if desired. Margins can even have negative values.

The width and height of each box is equal to the width and height of the outer margin box. Note that this is not the necessarily the same as the width and height of the content area.

A box can contain any number of other boxes, creating a hierarchy of boxes that corresponds to the nesting of page elements. The browser window serves as the root element for this hierarchy.

## Box Types

There are two basic types of boxes, block and inline. Block boxes are generated by elements such as P, DIV or TABLE. Inline boxes are generated by tags such as B, I or SPAN and actual content like text and images.

The box type may also be set using the display property. Setting a value of block on an inline element, for example, will cause it to be treated as a block element. Note that if you set the display to none, no box is created. The browser acts as if the element did not exist. Likewise, any nested elements are ignored as well, even if they specifically declare some other display value.

*There are other types of boxes which apply to special elements like lists and tables. However, these are ultimately treated as block or inline boxes for positioning purposes. As such, these other box types not covered here.*

## Positioning Types

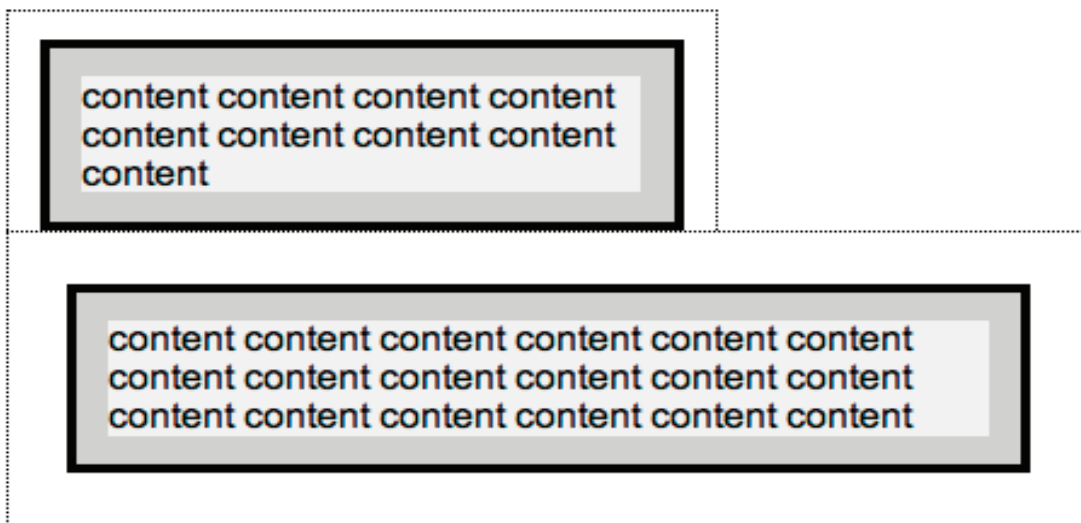
There are five positioning modes or schemes in CSS2.1: normal (The default), float, relative, fixed and absolute. Each has its own set of rules. Every box is positioned using one of these schemes but different boxes will use different schemes depending on their position and float style settings.

### Normal Flow

Normal flow is the default scheme used for positioning. It applies to any element that does not specify a 'position' and is not floated.

In this scheme, block boxes flow vertically starting at the top of their containing block with each placed directly below the preceding one. Inline boxes flow horizontally from left to right.

You should note that vertical margins are collapsed in the normal flow. That is, instead of adding the bottom margin of a box to the top margin of the one immediately below it, only the larger of the two values is used, as illustrated here.



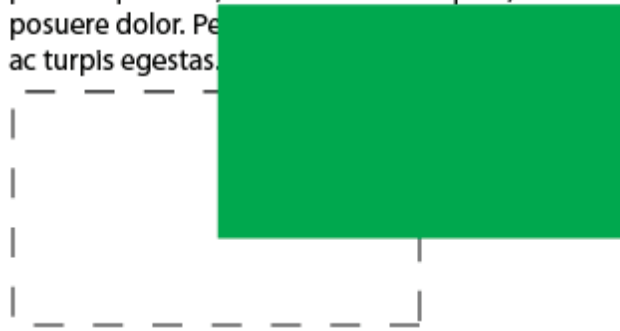
Horizontal margins, however, are never collapsed.

## Relative Positioning

When an element specifies `'position:relative'`; it is initially positioned following the normal flow rules. Surrounding boxes are positioned accordingly. Then, the box is moved according to its offset properties.

This can sometime cause what is sometimes referred to as ghosting. The box physically stays in the document flow, but visually is moved to the correct place.

Mauris faucibus. Aenean pulvinar ornare eros. Nam volutpat consequat  
est. Etiam turpis nisl, ultricies non, tincidunt eget, semper sit amet, erat. I  
Nam suscipit vehicula mauris. Cras tellus. In hac habitasse platea dictum.  
In turpis mattis ultrices. Donec tortor erat, adipiscing non, interdum at, Ir  
Mauris feugiat enim nec justo. Quisque blandit orci nec metus. Curabitur  
posuere posuere, est tellus laculis sapien, sed suscipit nunc eros In nunc.  
posuere dolor. Pe ac turpis egestas.



Example of ghosting

The offset values are specified using a combination of the top, right, left and bottom style properties. The value of each is interpreted as the distance the box's corresponding outer edge should be moved with respect to its original position in the normal flow.

Note that opposing offsets are constrained. For example, if you specify both left and right and the value of one is not the exact negative of the other, the right setting will be ignored. A specific width setting may also cause an offset to be ignored. The same is true of the top, bottom and height properties.

In practice, you'll probably want to specify only one of left and right and one of top and bottom.

## Absolute Positioning

This positioning scheme applies to any element that has its position property set to absolute or fixed.

Such boxes are removed from the normal flow and have no effect on boxes in that flow. Like floated elements, absolutely positioned elements are always treated as block-level elements. As such, they establish a new containing block for any descendants, i.e., any elements contained within the absolutely positioned element.

The position of an absolutely positioned element is determined by its offset values: top, right, bottom and left. These values work in much the same way as with relatively positioned elements.

But unlike relative positioning, where the offsets are measured from the element's position in the normal flow, an absolutely positioned element is offset from its container block.

The containing block of an absolutely positioned element is defined a little differently than it is for other elements. **The containing block for an absolutely positioned element is established by its closest, positioned ancestor.** That is, the nearest element outside it that has a position of absolute, relative or fixed. If there is no such ancestor element, the initial containing block (the browser window) is used.

Recall that non-absolutely positioned elements used the containing block of their closest, block-level ancestor. But for absolute elements the containing element can be an inline element.

Additionally, if that containing element is a block-level element, the padding edge of that element forms the container block, not the content edge. In other words, the offsets are measured from just inside the border of the containing element.

If the containing element is an inline-level element, it gets a little more complicated. Since an inline element may generate several line boxes, the container box is defined as the area bordered by the top and left content edges of the first box within that element and the bottom and right content edges of the last box it contains.