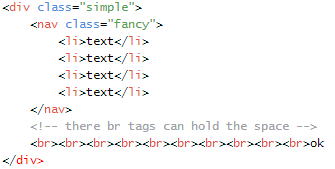
**position**

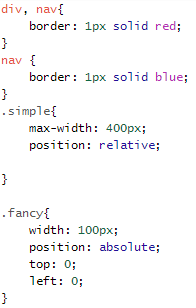
In order to make more complex layouts, we need to discuss the position property. It has a bunch of possible values, and their names make no sense and are impossible to remember. Let's go through them one by one, but maybe you should bookmark this page too.

* **Static**. This is the default for every single page element. Different elements don't have different default values for positioning, they all start out as static. Static doesn't mean much, it just means that the element will flow into the page as it normally would. The only reason you would ever set an element to position: static is to forcefully-remove some positioning that got applied to an element outside of your control. This is fairly rare, as positioning doesn't cascade.

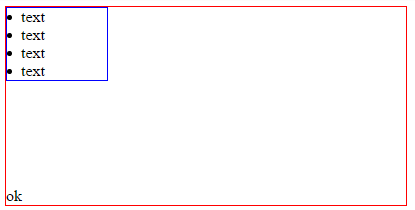
Html code:



CSS code:



Output:



In the above code, The nav element is absolutely positioned (so not in normal flow). The br element is statically positioned (the default) so it is in normal flow. The height of the container is calculated based on the height of the br elements (ignoring the height of the nav element entirely).

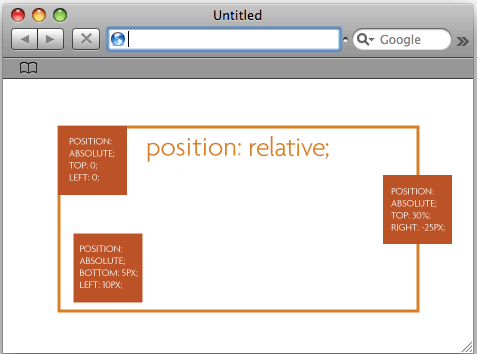
**Relative**. This type of positioning is probably the most confusing and misused. What it really means is "relative to itself". If you set position: relative; on an element but no other positioning attributes (top, left, bottom or right), it will no effect on it's positioning at all, it will be exactly as it would be if you left it as position: static; But if you DO give it some other positioning attribute, say, top: 10px;, it will shift it's position 10 pixels DOWN from where it would NORMALLY be. I'm sure you can imagine, the ability to shift an element around based on it's regular position is pretty useful. I find myself using this to line up form elements many times that have a tendency to not want to line up how I want them to.

There are two other things that happen when you set position: relative; on an element that you should be aware of. One is that it introduces the ability to use z-index on that element, which doesn't really work with statically positioned elements. Even if you don't set a z-index value, this element will now appear **on top** of any other statically positioned element. You can't fight it by setting a higher z-index value on a statically positioned element. The other thing that happens is it **limits the scope of absolutely positioned child elements**. Any element that is a child of the relatively positioned element can be absolutely positioned within that block. This brings up some powerful opportunities which is discussed below.

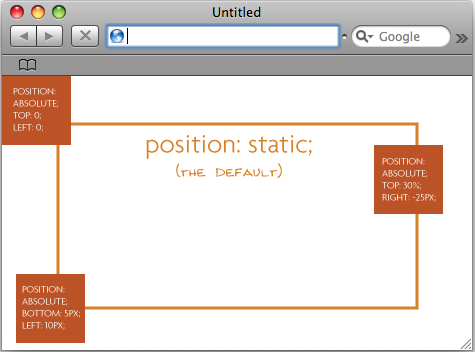
**Absolute Positioning Inside Relative Positioning**

A page element with**relative positioning**gives you the control to **absolutely position**children elements inside of it.

Here is a visual:



The relative positioning on the parent is the big deal here. Look what would happen if you forgot that:



Might not look like a big deal in this small example, but it really is a significant change. What is happening is the absolutely positioned elements are positioning themselves in relation to the **body** element instead of their direct parent. So if the browser window grows, that one in the bottom left is going to stick with the browser window, not hang back inside like his well behaved brother from the first image.

**Absolute**. This is a very powerful type of positioning that allows you to literally place any page element exactly where you want it. You use the positioning attributes top, left bottom and right to set the location. Remember that these values will be relative to the next parent element with relative (or absolute) positioning. If there is no such parent, it will default all the way back up to the <html> element itself meaning it will be placed relatively to the page itself.

The trade-off, and most important thing to remember, about absolute positioning is that these elements are removed from the flow of elements on the page.

*Absolutely positioned elements don't decide their own width or height, relative(or static) positioned elements don't decide their own height(only).*

Not only absolutely positioned element but all their descendants will be out of flow

**Fixed**. This type of positioning is fairly rare but certainly has its uses. A fixed position element is positioned relative to the viewport, or the browser window itself. The viewport doesn't change when the window is scrolled, so a fixed positioned element will stay right where it is when the page is scrolled, creating an effect a bit like the old school "frames" days.

**MDN documentation on position property**

The **position** [CSS](https://developer.mozilla.org/en-US/docs/CSS) property chooses alternative rules for positioning elements, designed to be useful for scripted animation effects.

Positioned elements have a higher z-index than non-positioned, making them stay in a higher layer (on top of).

A **positioned element** is an element whose [computed](https://developer.mozilla.org/en-US/docs/CSS/computed_value) position property is either; relative, absolute, fixed or sticky.

A **relatively positioned element** is an element whose [computed](https://developer.mozilla.org/en-US/docs/CSS/computed_value) position property is relative.

An **absolutely positioned element** is an element whose [computed](https://developer.mozilla.org/en-US/docs/CSS/computed_value) position property is absolute or fixed.

A **stickily positioned element** is an element whose [computed](https://developer.mozilla.org/en-US/docs/CSS/computed_value) position property is sticky.

The [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top), [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom), and [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left) properties specify the position of positioned elements.

## Syntax

/\* Keyword values \*/

position: static;

position: relative;

position: absolute;

position: fixed;

position: sticky;

/\* Global values \*/

position: inherit;

position: initial;

position: unset;

### Values

static

This keyword lets the element use the normal behavior, that is it is laid out in its current position in the flow.  The top, right, bottom, left and z-index properties do not apply.

relative

This keyword lays out all elements as though the element were not positioned, and then adjust the element's position, without changing layout (and thus leaving a gap for the element where it would have been had it not been positioned). The effect of position:relative on table-\*-group, table-row, table-column, table-cell, and table-caption elements is undefined.

absolute

Do not leave space for the element. Instead, position it at a specified position relative to its closest positioned ancestor or to the containing block. Absolutely positioned boxes can have margins, they do not collapse with any other margins.

fixed

Do not leave space for the element. Instead, position it at a specified position relative to the screen's viewport and don't move it when scrolled. When printing, position it at that fixed position on every page. This value always create a new stacking context.

sticky 

The box position is calculated according to the normal flow (this is called the position in normal flow). Then the box is offset relative to its flow root and containing block and in all cases, including table elements, does not affect the position of any following boxes. When a box B is stickily positioned, the position of the following box is calculated as though B were not offset. The effect of ‘position: sticky’ on table elements is the same as for ‘position: relative’.

### Formal syntax

[How to read CSS syntax.](https://developer.mozilla.org/docs/Web/CSS/Value_definition_syntax)

static [|](https://developer.mozilla.org/en-US/docs/CSS/Value_definition_syntax#Single_bar) relative [|](https://developer.mozilla.org/en-US/docs/CSS/Value_definition_syntax#Single_bar) absolute [|](https://developer.mozilla.org/en-US/docs/CSS/Value_definition_syntax#Single_bar) sticky [|](https://developer.mozilla.org/en-US/docs/CSS/Value_definition_syntax#Single_bar) fixed

## Examples

### Relative positioning

#### HTML Content

<div class="box" id="one">One</div>

<div class="box" id="two">Two</div>

<div class="box" id="three">Three</div>

<div class="box" id="four">Four</div>

#### CSS Content

.box {

display: inline-block;

background: red;

width: 100px;

height: 100px;

float: left;

margin: 20px;

color: white;

}

#two {

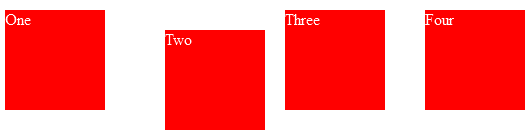
position: relative;

top: 20px;

left: 20px;

}

Note how the other elements are displayed as if "Two" were in its normal position and taking up space.



### Absolute positioning

Elements that are positioned relatively are still considered to be in the normal flow of elements in the document. In contrast, an element that is positioned absolutely is taken out of the flow and thus takes up no space when placing other elements. The absolutely positioned element is positioned relative to nearest positioned ancestor(non static). If a positioned ancestor doesn't exist, the initial container is used.

In the example below, a positioned ancestor doesn't exist and box Two is positioned absolutely relative to initial container:

#### HTML Content

<div class="box" id="one">One</div>

<div class="box" id="two">Two</div>

<div class="box" id="three">Three</div>

<div class="box" id="four">Four</div>

#### CSS Content

.box {

display: inline-block;

background: red;

width: 100px;

height: 100px;

float: left;

margin: 20px;

color: white;

}

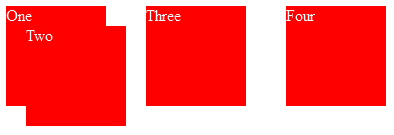
#two {

position: absolute;

top: 20px;

left: 20px;

}



If box One had not been positioned relative, it would not be the nearest positioned ancestor, and so box Two would have appeared relative to the upper left corner of the outer box instead.

### Fixed positioning

Fixed positioning is similar to absolute positioning, with the exception that the element's containing block is the viewport. This is often used to create a floating element that stays in the same position even after scrolling the page. In the example below the "One" box is fixed 80px from the top of the page and 20px from the left:

#### HTML Content

<div class="outer">

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam congue tortor eget pulvinar lobortis.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nam ac dolor augue.

Pellentesque mi mi, laoreet et dolor sit amet, ultrices varius risus. Nam vitae iaculis elit.

Aliquam mollis interdum libero. Sed sodales placerat egestas. Vestibulum ut arcu aliquam purus viverra dictum vel sit amet mi.

Duis nisl mauris, aliquam sit amet luctus eget, dapibus in enim. Sed velit augue, pretium a sem aliquam, congue porttitor tortor.

Sed tempor nisl a lorem consequat, id maximus erat aliquet. Sed sagittis porta libero sed condimentum.

Aliquam finibus lectus nec ante congue rutrum. Curabitur quam quam, accumsan id ultrices ultrices, tempor et tellus.

</p>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam congue tortor eget pulvinar lobortis.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nam ac dolor augue.

Pellentesque mi mi, laoreet et dolor sit amet, ultrices varius risus. Nam vitae iaculis elit.

Aliquam mollis interdum libero. Sed sodales placerat egestas. Vestibulum ut arcu aliquam purus viverra dictum vel sit amet mi.

Duis nisl mauris, aliquam sit amet luctus eget, dapibus in enim. Sed velit augue, pretium a sem aliquam, congue porttitor tortor.

Sed tempor nisl a lorem consequat, id maximus erat aliquet. Sed sagittis porta libero sed condimentum.

Aliquam finibus lectus nec ante congue rutrum. Curabitur quam quam, accumsan id ultrices ultrices, tempor et tellus.

</p>

<div class="box" id="one">One</div>

</div>

#### CSS Content

.box {

background: red;

width: 100px;

height: 100px;

margin: 20px;

color: white;

}

#one {

position: fixed;

top: 80px;

left: 10px;

}

.outer {

width: 500px;

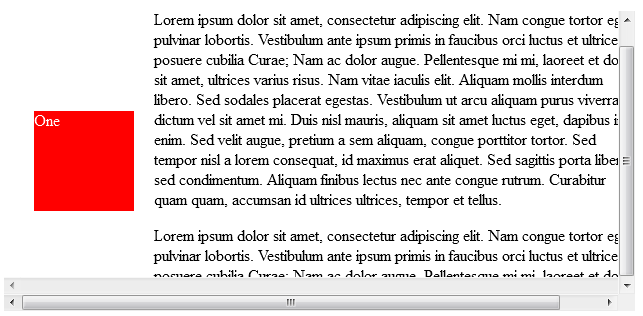
height: 300px;

overflow: scroll;

padding-left: 150px;

}

When viewing the top of the page, the position box appears in the upper left, and after scrolling, it remains in the same place relative to the viewport:



### Sticky positioning

Sticky positioning is a hybrid of relative and fixed positioning.  The element is treated as relative positioned until it crosses a specified threshold, at which point it is treated as fixed positioned.  For instance:

#one { position: sticky; top: 10px; }

will behave just like a relatively positioned element until the viewport scrolls such that the element would be less than 10px from the top.  Then, it will be fixed to 10px from the top until the viewport is scrolled back past this threshold.

Sticky positioning is commonly used for the headings in an alphabetized listing.  The B heading will appear just below the items that begin with A until they are scrolled offscreen.  Rather than sliding offscreen with the rest of the content, the B heading will then remain fixed to the top of the viewport until all the B items have scrolled offscreen, at which point it will be covered up by the C heading.

You must specify a threshold with at least one of [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top), [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom), or [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left) for sticky positioning to behave as expected.  Otherwise, it will be indistinguishable from relative positioning.

#### HTML Content

<div>

<dl>

<dt>A</dt>

<dd>Andrew W.K.</dd>

<dd>Apparat</dd>

<dd>Arcade Fire</dd>

<dd>At The Drive-In</dd>

<dd>Aziz Ansari</dd>

</dl>

<dl>

<dt>C</dt>

<dd>Chromeo</dd>

<dd>Common</dd>

<dd>Converge</dd>

<dd>Crystal Castles</dd>

<dd>Cursive</dd>

</dl>

<dl>

<dt>E</dt>

<dd>Explosions In The Sky</dd>

</dl>

<dl>

<dt>T</dt>

<dd>Ted Leo & The Pharmacists</dd>

<dd>T-Pain</dd>

<dd>Thrice</dd>

<dd>TV On The Radio</dd>

<dd>Two Gallants</dd>

</dl>

</div>

#### CSS Content

dl {

margin: 0;

padding: 24px 0 0 0;

}

dt {

background: #B8C1C8;

border-bottom: 1px solid #989EA4;

border-top: 1px solid #717D85;

color: #FFF;

font: bold 18px/21px Helvetica, Arial, sans-serif;

margin: 0;

padding: 2px 0 0 12px;

position: -webkit-sticky;

position: sticky;

width: 99%;

top: 0px;

}

dd {

font: bold 20px/45px Helvetica, Arial, sans-serif;

margin: 0;

padding: 0 0 0 12px;

white-space: nowrap;

}

dd + dd {

border-top: 1px solid #CCC

}



---------------------------------------------------------------------------------------------------------------------------------





For relatively positioned elements, the [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top) or [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom) property specifies the vertical offset from the normal position and the [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left) or [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right) property specifies the horizontal offset.

For absolutely positioned elements, the [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top), [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom), and [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left) properties specify offsets from the edge of the element's containing block (what the element is positioned relative to). The margin of the element is then positioned inside these offsets.

Most of the time, absolutely positioned elements have auto values of [height](https://developer.mozilla.org/en-US/docs/Web/CSS/height) and [width](https://developer.mozilla.org/en-US/docs/Web/CSS/width) computed to fit the contents of the element. However, non-replaced absolutely positioned elements can be made to fill the available space by specifying (as other than auto) both [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top) and [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom) and leaving [height](https://developer.mozilla.org/en-US/docs/Web/CSS/height) unspecified (that is, auto). Likewise for [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left), [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right), and [width](https://developer.mozilla.org/en-US/docs/Web/CSS/width).

Except for the case just described of absolutely positioned elements filling the available space:

* If both [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top) and [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom) are specified (technically, not auto), [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top) wins.
* If both [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left) and [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right) are specified, [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left) wins when [direction](https://developer.mozilla.org/en-US/docs/Web/CSS/direction) is ltr (English, horizontal Japanese, etc.) and [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right) wins when [direction](https://developer.mozilla.org/en-US/docs/Web/CSS/direction) is rtl (Persian, Arabic, Hebrew, etc.).

A relatively positioned element is positioned to itself. An absolutely positioned element will keep the flow of elements (unless specified using left right etc) but does not affect elements after it.

## Relationships between 'display', 'position', and 'float'

The three properties that affect box generation and layout — ['display'](http://www.w3.org/TR/CSS21/visuren.html#propdef-display), ['position'](http://www.w3.org/TR/CSS21/visuren.html#propdef-position), and ['float'](http://www.w3.org/TR/CSS21/visuren.html#propdef-float) — interact as follows:

1. If ['display'](http://www.w3.org/TR/CSS21/visuren.html#propdef-display) has the value 'none', then ['position'](http://www.w3.org/TR/CSS21/visuren.html#propdef-position) and ['float'](http://www.w3.org/TR/CSS21/visuren.html#propdef-float) do not apply. In this case, the element generates no box.
2. Otherwise, if ['position'](http://www.w3.org/TR/CSS21/visuren.html#propdef-position) has the value 'absolute' or 'fixed', the box is absolutely positioned, the computed value of ['float'](http://www.w3.org/TR/CSS21/visuren.html#propdef-float) is 'none', and display is set according to the table below. The position of the box will be determined by the ['top'](http://www.w3.org/TR/CSS21/visuren.html#propdef-top), ['right'](http://www.w3.org/TR/CSS21/visuren.html#propdef-right), ['bottom'](http://www.w3.org/TR/CSS21/visuren.html#propdef-bottom) and ['left'](http://www.w3.org/TR/CSS21/visuren.html#propdef-left) properties and the box's containing block.
3. Otherwise, if 'float' has a value other than 'none', the box is floated and 'display' is set according to the table below.
4. Otherwise, if the element is the root element, 'display' is set according to the table below, except that it is undefined in CSS 2.1 whether a specified value of 'list-item' becomes a computed value of 'block' or 'list-item'.
5. Otherwise, the remaining ['display'](http://www.w3.org/TR/CSS21/visuren.html#propdef-display) property values apply as specified.

|  |  |
| --- | --- |
| **Specified value** | **Computed value** |
| inline-table | table |
| inline, table-row-group, table-column, table-column-group, table-header-group, table-footer-group, table-row, table-cell, table-caption, inline-block | block |
| others | same as specified |