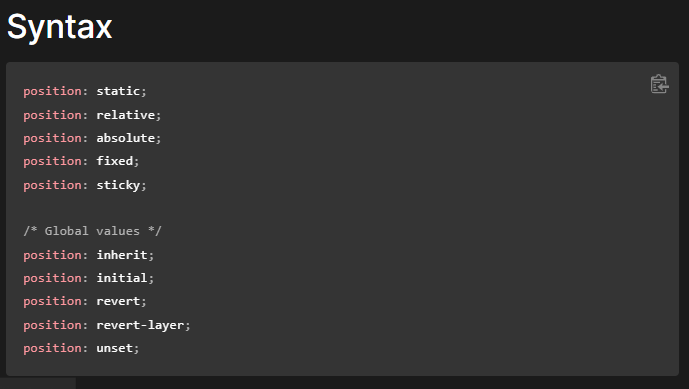
Position

The position CSS property sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements.

Position values:

1. static
2. relative
3. fixed
4. absolute
5. sticky



Values

static

The element is positioned according to the normal flow of the document. The top, right, bottom, left, and z-index properties have no effect. This is the default value.

relative

The element is positioned according to the normal flow of the document, and then offset relative to itself based on the values of top, right, bottom, and left. The offset does not affect the position of any other elements; thus, the space given for the element in the page layout is the same as if position were static.

This value creates a new stacking context when the value of z-index is not auto. Its effect on table-\*-group, table-row, table-column, table-cell, and table-caption elements is undefined.

absolute

The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to its closest positioned ancestor, if any; otherwise, it is placed relative to the initial containing block. Its final position is determined by the values of top, right, bottom, and left.

This value creates a new stacking context when the value of z-index is not auto. The margins of absolutely positioned boxes do not collapse with other margins.

fixed

The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to the initial containing block established by the viewport, except when one of its ancestors has a transform, perspective, or filter property set to something other than none (see the CSS Transforms Spec), or the will-change property is set to transform, in which case that ancestor behaves as the containing block. (Note that there are browser inconsistencies with perspective and filter contributing to containing block formation.) Its final position is determined by the values of top, right, bottom, and left.

This value always creates a new stacking context. In printed documents, the element is placed in the same position on every page.

sticky

The element is positioned according to the normal flow of the document, and then offset relative to its nearest scrolling ancestor and containing block (nearest block-level ancestor), including table-related elements, based on the values of top, right, bottom, and left. The offset does not affect the position of any other elements.

This value always creates a new stacking context. Note that a sticky element "sticks" to its nearest ancestor that has a "scrolling mechanism" (created when overflow is hidden, scroll, auto, or overlay), even if that ancestor isn't the nearest actually scrolling ancestor.

### [Types of positioning](https://developer.mozilla.org/en-US/docs/Web/CSS/position#types_of_positioning)

* A **positioned element** is an element whose [computed](https://developer.mozilla.org/en-US/docs/Web/CSS/computed_value) position value is either relative, absolute, fixed, or sticky. (In other words, it's anything except static.)
* A **relatively positioned element** is an element whose [computed](https://developer.mozilla.org/en-US/docs/Web/CSS/computed_value) position value is relative. The [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top) and [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom) properties specify the vertical offset from its normal position; the [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left) and [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right) properties specify the horizontal offset.
* An **absolutely positioned element** is an element whose [computed](https://developer.mozilla.org/en-US/docs/Web/CSS/computed_value) position value is absolute or fixed. 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 edges of the element's [containing block](https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block). (The containing block is the ancestor relative to which the element is positioned.) If the element has margins, they are added to the offset. The element establishes a new [block formatting context](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Block_formatting_context) (BFC) for its contents.
* A **stickily positioned element** is an element whose [computed](https://developer.mozilla.org/en-US/docs/Web/CSS/computed_value) position value is sticky. It's treated as relatively positioned until its [containing block](https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block) crosses a specified threshold (such as setting [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top) to value other than auto) within its flow root (or the container it scrolls within), at which point it is treated as "stuck" until meeting the opposite edge of its [containing block](https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block).

Example:

<!DOCTYPE html>

<html>

<style>

section{

width:80%;

border:2px solid black;

margin:2em;

height:100vh;

}

.content{

margin:1em;

width:25%;

height:8vh;

text-align:center;

border-radius:10px;

padding:1rem;

}

.box1{

background-color:red;

color:white;

position:relative;

top:10px;

left:50px;

}

.box2{

background-color:blue;

color:white;

position:absolute;

top:80px;

left:50px;

}

.box3{

background-color:orange;

color:white;

position:fixed;

top:80px;

left:50px;

}

.box4{

background-color:green;

color:white;

position:sticky;

}

</style>

<body>

<section>

<div class="box1 content">Relative</div>

<div class="box2 content">Absolute</div>

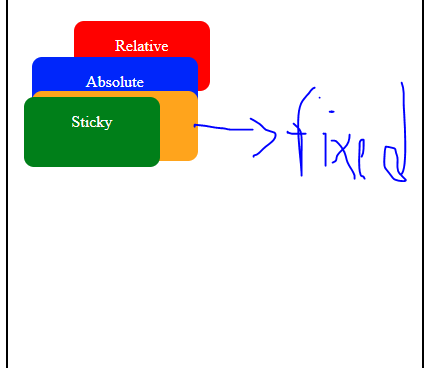
<div class="box3 content">Fixed</div>

<div class="box4 content">Sticky</div>

</section>

</body>

</html>



1. **Static** - this is the default value, all elements are in order as they appear in the document.
2. **Relative** - the element is positioned relative to its normal position.
3. **Absolute** - the element is positioned absolutely to its first positioned parent.
4. **Fixed** - the element is positioned related to the browser window.
5. **Sticky** - the element is positioned based on the user’s scroll position.

## What Is Relative Positioning?

When you set the position relative to an element, without adding any other positioning attributes (top, bottom, right, left) **nothing will happen**. When you add an additional position apart from relative, such as left: 20px the element will move 20px to the right from its normal position. Here, you can see that **this element is relative to itself**. When the element moves, no other element on the layout will be affected.

There is a thing you should keep in mind while setting position - relative to an element limits the scope of absolutely positioned child elements. This means that any element that is the child of this element can be absolutely positioned within this block.

After this brief explanation, we need to back it up, by showing an example.

In this example, you will see how the relative positioned element moves when its attributes are changed. The first element moves to the **left** and **top** from its normal position, while **the second element stays in the same place** because none of the additional positioning attributes were changed.

### HTML

<div id=”first\_element”>First element</div>

<div id=”second\_element”>Second element</div>

### CSS

#first\_element {

position: relative;

left: 30px;

top: 70px;

width: 500px;

background-color: #fafafa;

border: solid 3px #ff7347;

font-size: 24px;

text-align: center;

}

#second\_element {

position: relative;

width: 500px;

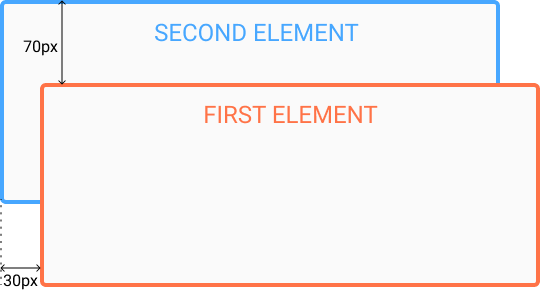
background-color: #fafafa;

border: solid 3px #ff7347;

font-size: 24px;

text-align: center;

}



## **What Is Absolute Positioning?**

Absolute positioning allows you to **place your element precisely where you want it**.

Absolute positioning is done **relative to the first relatively (or absolutely) positioned parent element**. In the case when there is no positioned parent element, the element that has position set to absolute will be positioned related **directly to the HTML element (the page itself)**.

An important thing to keep in mind while using absolute positioning is to make sure it is **not overused**, otherwise, it can lead to a maintenance nightmare.

The next thing, yet again, is to show an example of absolute positioning.

In the example, the parent element has the position set to relative. Now, when you set the position of the child element to absolute, **any additional positioning will be done relative to the parent element**. The child element moves relatively to the top of the parent element by 100px and right of the parent element by 40px.

### HTML

<div id=”parent”>

<div id=”child”></div>

</div>

### CSS

#parent {

position: relative;

width: 500px;

height: 400px;

background-color: #fafafa;

border: solid 3px #9e70ba;

font-size: 24px;

text-align: center;

}

#child {

position: absolute;

right: 40px;

top: 100px;

width: 200px;

height: 200px;

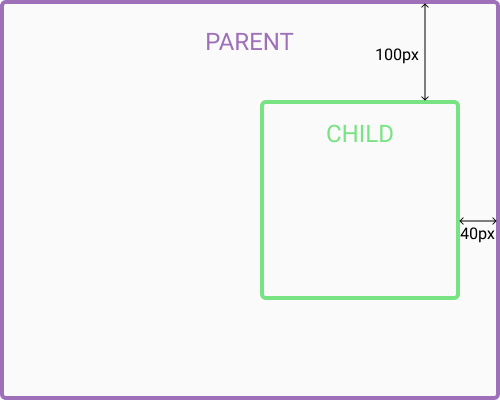
background-color: #fafafa;

border: solid 3px #78e382;

font-size: 24px;

text-align: center;

}



Through these examples, you have seen differences between absolute and relative positioned elements.

Program example:

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>CSS Position</title>

<link rel="stylesheet" href="css/style.css">

</head>

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: "Roboto", sans-serif;

font-size: 1.5rem;

min-height: 200vh;

}

.outer-container {

border: 3px dashed #000;

width: 75vw;

height: 85vh;

margin: 40px auto;

position: relative;

}

.inner-container {

border: 2px solid #00f;

width: 40vw;

height: 50vh;

margin: 200px auto;

}

.box {

width: 150px;

height: 150px;

color: #fff;

padding: 1rem;

}

.absolute {

background-color: #00f;

position: absolute;

top: 0;

left: 0;

z-index: 1;

}

.relative {

background-color: #f00;

position: relative;

top: 300px;

left: 100px;

}

.fixed {

background-color: green;

position: fixed;

top: 100px;

}

.sticky {

background-color: #000;

position: sticky;

top: 0;

}

/\* Remember, how to make things disappear! \*/

/\*//////////////////////////\*/

html { scroll-behavior: smooth; }

section {

height: 100vh;

}

.blue { background-color: #00f;}

.red { background-color: #f00;}

.green { background-color: green;}

header, footer {

color: #fff;

text-align: center;

height: 100px;

}

header {

position: sticky;

top: 0;

font-size: 5rem;

}

footer {

background-color: #000;

position: sticky;

bottom: 0;

font-size: 3rem;

}

a:visited {

color: #fff;

}

.social {

background-color: royalblue;

color: #fff;

font-size: inherit;

padding: 1rem;

position: fixed;

top: 30%;

left: 0;

z-index: 1;

}

</style>

<body>

<div class="outer-container">

<div class="inner-container">

<div class="box absolute">

<p>Absolute</p>

</div>

<div class="box relative">

<p>Relative</p>

</div>

<div class="box fixed">

<p>Fixed</p>

</div>

<div class="box sticky">

<p>Sticky</p>

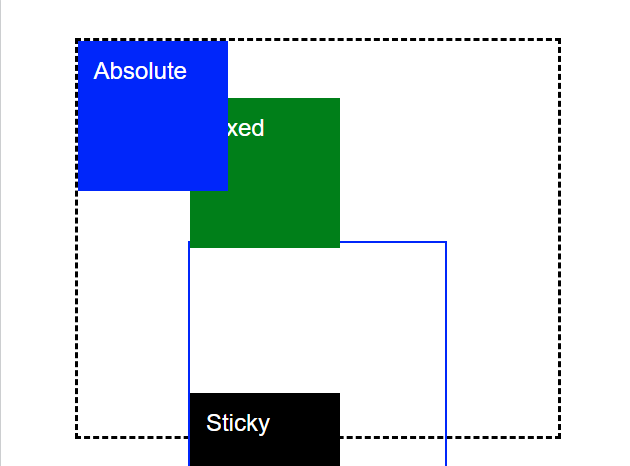
</div>

</div>

</div>

</body>

</html>



Example program - IMP

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>CSS Position</title>

<link rel="stylesheet" href="css/style.css">

</head>

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: "Roboto", sans-serif;

font-size: 1.5rem;

min-height: 200vh;

}

.outer-container {

border: 3px dashed #000;

width: 75vw;

height: 85vh;

margin: 40px auto;

position: relative;

}

.inner-container {

border: 2px solid #00f;

width: 40vw;

height: 50vh;

margin: 200px auto;

}

.box {

width: 150px;

height: 150px;

color: #fff;

padding: 1rem;

}

.absolute {

background-color: #00f;

position: absolute;

top: 0;

left: 0;

z-index: 1;

}

.relative {

background-color: #f00;

position: relative;

top: 300px;

left: 100px;

}

.fixed {

background-color: green;

position: fixed;

top: 100px;

}

.sticky {

background-color: #000;

position: sticky;

top: 0;

}

/\* Remember, how to make things disappear! \*/

/\*//////////////////////////\*/

html { scroll-behavior: smooth; }

section {

height: 100vh;

}

.blue { background-color: #00f;}

.red { background-color: #f00;}

.green { background-color: green;}

header, footer {

color: #fff;

text-align: center;

height: 100px;

}

header {

position: sticky;

top: 0;

font-size: 5rem;

}

footer {

background-color: #000;

position: sticky;

bottom: 0;

font-size: 3rem;

}

a:visited {

color: #fff;

}

.social {

background-color: royalblue;

color: #fff;

font-size: inherit;

padding: 1rem;

position: fixed;

top: 30%;

left: 0;

z-index: 1;

}

</style>

<body>

<button class="social">🚀</button>

<section id="one">

<header class="blue">Header One</header>

<h2>One</h2>

</section>

<section id="two">

<header class="red">Header Two</header>

<h2>Two</h2>

</section>

<section id="three">

<header class="green">Header Three</header>

<h2>Three</h2>

</section>

<footer>

<a href="#one">One</a> |

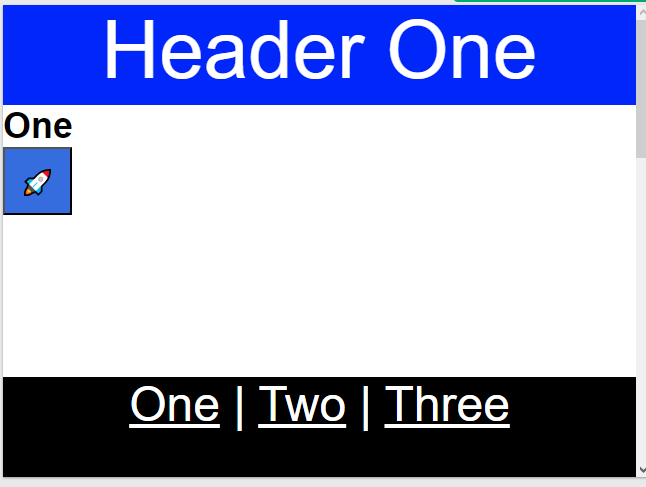
<a href="#two">Two</a> |

<a href="#three">Three</a>

</footer>

</body>

</html>



For reference: <https://kolosek.com/css-position-relative-vs-position-absolute/#:~:text=Relative%20-%20the%20element%20is%20positioned,on%20the%20user%27s%20scroll%20position>.