Absolute, Relative, Fixed Positioning: How Do They Differ?

https://css-tricks.com/absolute-relative-fixed-positioining-how-do-they-differ/

Chris Coyier Oct 14, 2008 Aug 13, 2020

Let's talk about the `position` property (https://css-tricks.com/almanac/properties/p/position/) . I know beginners are curious about this. Here's a question I got recently:

> *I am fairly new to web design, and haven't mastered the differences in positioning of elements. I know there is `absolute`, `fixed`, and `relative`. Are there any others? Also, do they majorly differ? And when should you use which?*

(#aa-short-answer) Short answer

There are two more: `static`, which is the default, and `sticky`, which is a whole fancy thing. **Yes**, all of these majorly differ! Each of them is incredibly useful and which you should use of course depends on the desired result.

(#aa-longer-answer) Longer answer

An important concept to understand first is that every single element on a web page is a block. Literally a rectangle of pixels. This is easy to understand when you set the element to `display: block;` or if that element is block-level by default like a `<div>`. This means you can set a `width` and a `height` and that element will respect that. But elements that are `display:`

`inline;`, like a `<span>` by default, are *also* rectangles, they just flow onto the page differently, lining up horizontally as they can.

Now that you are picturing every single page element as a block of pixels, we can talk about how positioning is used to get the blocks of pixels exactly where you want them to go.

```css
.el {
  position: static;
  position: relative;
  position: absolute;
  position: fixed;
  position: sticky;
  position: inherit;
}
```

(#aa-static) 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.

(#aa-relative) 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 have 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 its 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 its 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 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 I talk about here (https://css-tricks.com/absolute-positioning-inside-relative-positioning/) .

(#aa-absolute) `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 relative 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. An element with this type of positioning is not affected by other elements and it doesn't affect other elements. This is a serious thing to consider every time you use absolute positioning. Its overuse or improper use can limit the flexibility of your site.

(#aa-fixed) `fixed`

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.

This might be used for something like a navigation bar that you want to remain visible at all times regardless of the pages scroll position. The concern with fixed positioning is that it can cause situations where the fixed element overlaps content such that is is inaccessible. The trick is having enough space to avoid that, and tricks like this (https://css-tricks.com/fixed-headers-and-jump-links-the-solution-is-scroll-margin-top/) .

(#aa-sticky) `sticky`

Sticky positioning is really unique! A `sticky` element will just sit there like a static element, but as you scroll *past* it, if it's parent element has room (usually: extra height) the sticky element will behave as if it's `fixed` until that parent element is out of room. It sounds weird in words like that, but it's easy to see what's happening in a demo (https://css-tricks.com/position-sticky-2/) .

(#aa-related-concepts) Related Concepts

- Layout like float (https://css-tricks.com/all-about-floats/) , flexbox (https://css-tricks.com/snippets/css/a-guide-to-flexbox/) , and grid (https://css-tricks.com/snippets/css/complete-guide-grid/) .
- The Box Model (https://css-tricks.com/the-css-box-model/)