Elad Shechter Follow

Jan 26, 2019 · 5 min read · Listen

CSS Position Sticky - How It Really Works!

CSS position sticky has really good browser support, yet most developers aren't using it.

The reason for that is twofold: First, the long wait for good browser support: It took quite a long time for browser support to happen, and by the time it did the feature was forgotten.

The second reason is that most developers don't fully understand the logic behind how it works, and that's where I come in.

I assume you all know CSS Positions, but let’s have a brief review:

Up until 3 years ago, there were four CSS Positions: `static`, `relative`, `absolute` and `fixed`.

The main difference between `static` or `relative` and `absolute` or `fixed` was the space they take up in the flow of the DOM. Positions `static` & `relative` keep their natural space in the flow of the document, while the `absolute` & `fixed` don’t — their space is removed and they have a floating behavior.

The new Position Sticky has similarities with all types, as I will soon explain.

My First Play with Position Sticky

I assume most of you have played with position sticky. So had I for quite a while, until I realized that I didn’t totally understand it.

On the first play with position sticky, everyone quickly understands that the element sticks when the viewport gets to the defined position.

Example:

```
.some-component{
    position: sticky;
    top: 0;
}
```

The problem is that sometimes it works while other times it doesn’t. When it does work, the element sticks, but in other parts of the scrolling it stops sticking. As someone who eats and breaths CSS daily, it was unacceptable for me to not understand the issue to its root , so I decided to thoroughly explore **Position Sticky**.

The Sticky Exploration

While playing with it, I quickly noticed that when an element with a `position: sticky` style is wrapped, and it is the only element inside the wrapper element, this element, which was defined `position: sticky` will not stick.

```
<!-- DOESN'T WORK!!! -->
<style>
    .sticky{
        position: sticky;
        top: 0;
    }
</style>

<div class="wrapper">
   <div class="sticky">
        SOME CONTENT
   </div>
</div>
```

When I added more elements inside the wrapper element, it started working.

Why is that?
The reason for that is that when an element is given the `position: sticky` style, the sticky item's container is the only area in which the sticky item can

stick. The item doesn’t have any elements to float over since it can only float over sibling elements, which, being an only child, it doesn’t have.

How CSS Position Sticky Really Works!

CSS position sticky has two main parts, **sticky item** & **sticky container.**

Sticky Item — is the element that we defined with the `position: sticky` styles. The element will float when the viewport position matches the position definition, for example: `top: 0px` .

Example

```
.some-component{
    position: sticky;
    top: 0px;
}
```

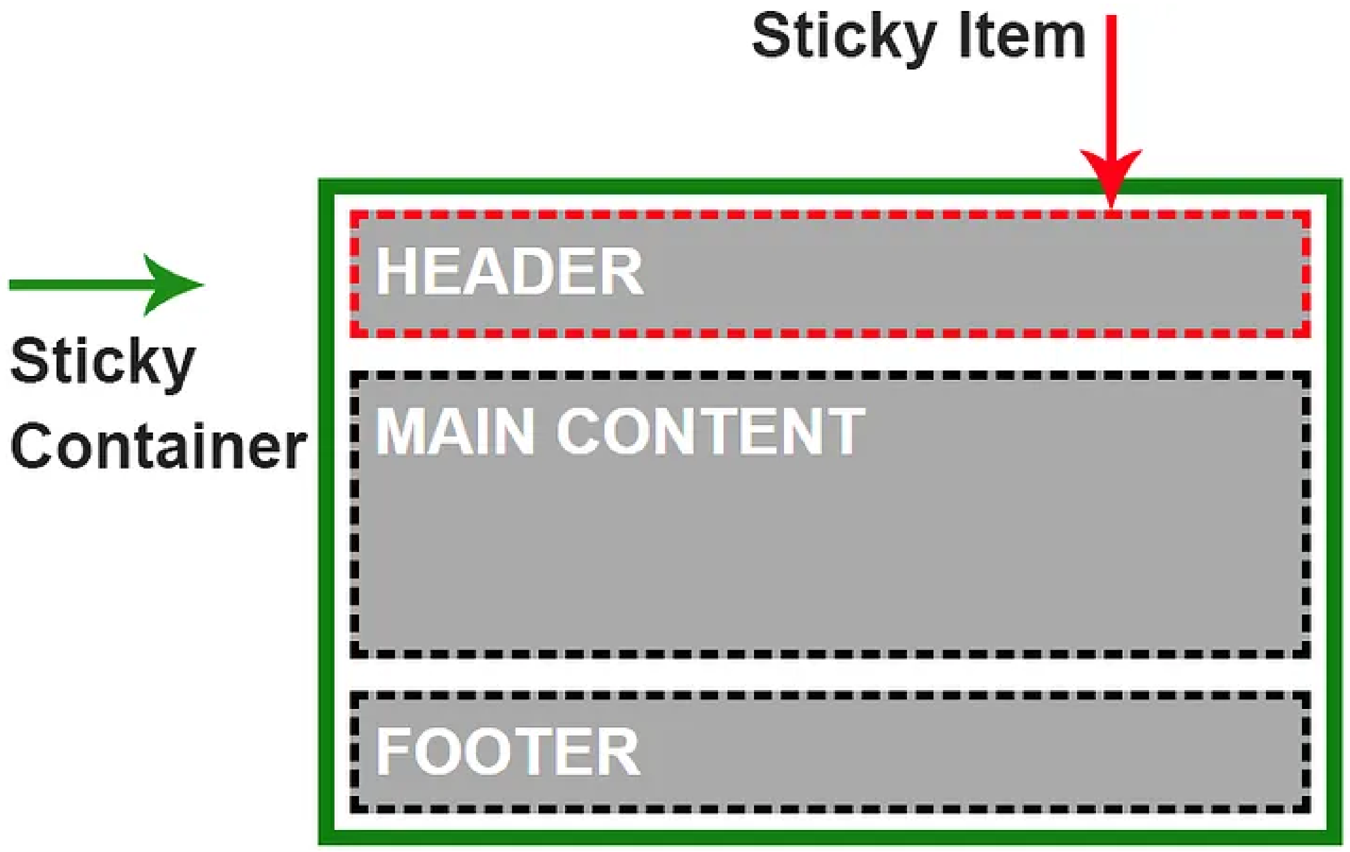
Sticky Container —is the HTML element which wraps the **sticky item.** This is the maximum area that the sticky item can float in.

When you define an element with `position: sticky` you’re automatically defining the parent element as a sticky container!

This is very important to remember! The container is the scope of the sticky item, and the item can't get out of its sticky container.

This is the reason why in the previous example, the sticky item didn't stick in the first place: The sticky item was the only child element in a sticky container.

Visual Example of CSS Position Sticky:



Sticky item & Sticky container

CodePen Example:

Understanding the CSS Sticky Behavior

Like I said before, CSS Position Sticky behaves differently than all the other CSS Positions, but on the other hand it does have some similarities with them. Let me explain:

Relative (or Static)— the stickily positioned element is similar to the relative and static positions because it keeps the natural gap in the DOM (stays in the flow).
Fixed— when the item sticks, it behaves exactly like `position: fixed`, floating in the same position of the view-port, removed from the flow.
Absolute — at the end of the sticking area, the element stops and stacks on top of the other element, much like an absolutely positioned element behaves inside a `position: relative` container.

Stick to the bottom?!

In most cases, you will use position sticky in order to stick an element to the top, something like this:

```
.component{
    position: sticky;
    top: 0;
}
```

This is exactly the situation it was created for, while before this, it could only be done using JavaScript.

But you can also use it to stick elements to the bottom. That means that the footer can be defined to have a sticky position, and it will always appear to stick to the bottom when scrolling down. When we reach the end of the sticky container, the element will stop in its natural position. It's better to use it on elements whose natural position is the bottom of the sticky container.

Full Example:

HTML

```
<main class="main-container">
  <header class="main-header">HEADER</header>
  <div class="main-content">MAIN CONTENT</div>
  <footer class="main-footer">FOOTER</footer>
</main>
```

CSS

```
.main-footer{
     position: sticky;
     bottom: 0;
}
```

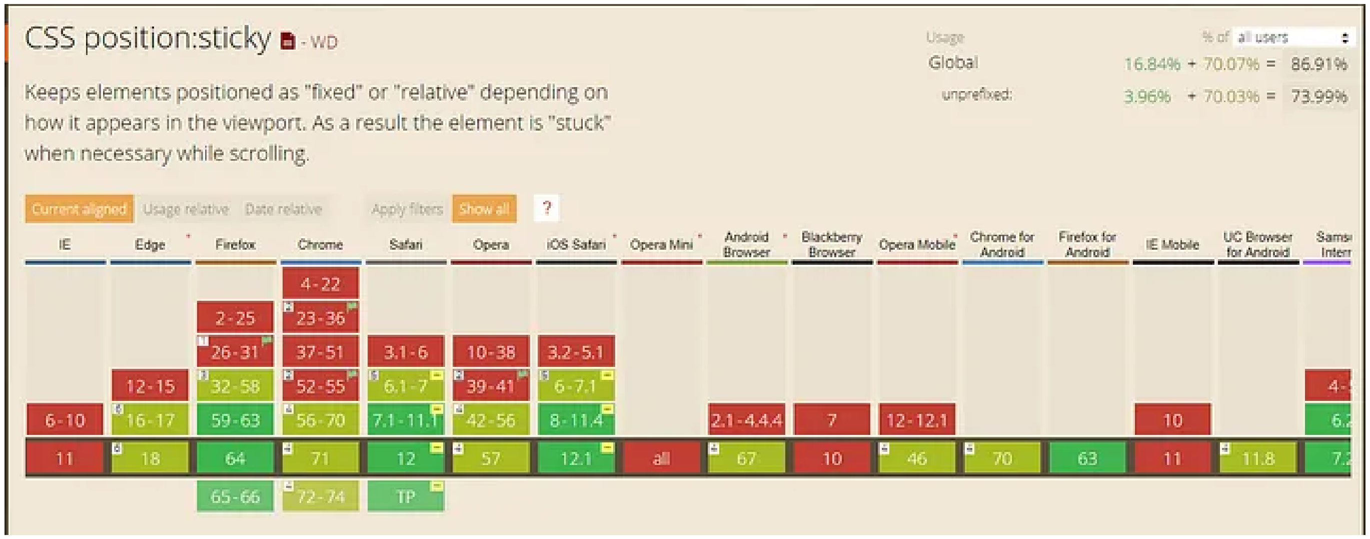
Live CodePen Example:

I use it in real life for sticky summary tables, and I can think of sticky footer navigation, which can work very nicely using this method.

Browsers Support

- Position Sticky is supported by all major modern browsers, except for old IE.
- For Safari browsers you will need to add the `-webkit` prefix.

```
position: -webkit-sticky; /* Safari */
position: sticky;
```



CSS position:sticky - WD

Keeps elements positioned as "fixed" or "relative" depending on how it appears in the viewport. As a result the element is "stuck" when necessary while scrolling.

Usage % of all users
Global 16.84% + 70.07% = 86.91%
unprefixed: 3.96% + 70.03% = 73.99%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Sams Interr
			4-22												
		2-25	23-36												
		26-31	37-51	3.1-6	10-38	3.2-5.1									
	12-15	32-58	52-55	6.1-7	39-41	6-7.1									4-
6-10	16-17	59-63	56-70	7.1-11.1	42-56	8-11.4		2.1-4.4.4	7	12-12.1			10		6.2
11	18	64	71	12	57	12.1	all	67	10	46	70	63	11	11.8	7.2
		65-66	72-74	TP											

can i use position sticky — more than 86% browser support

Final Words

That’s all,
I hope you’ve enjoyed this article and learned from my experience.
If you like this post, I would appreciate applause and sharing :-)

More of my CSS articles:
Why CSS HSL Colors are Better!
New CSS Logical Properties!
Becoming a CSS Grid Ninja!
The New Responsive Design Evolution

Who Am I?
I am Elad Shechter, a Web Developer specializing in CSS & HTML design and architecture. I work at Investing.com.

You can follow me via **Twitter**, and find more of my stuff on **my website**.

Who Am I?

- I'm CSS/HTML Architect at Investing.com (3/2018 - now)
- Last 6 Years worked at Walla! (2012 - 2/2018)
- Founder of CSS Masters Israel Facebook Group (June 2013)
- Writer at medium.com/@elad

You Can find me in my Facebook groups:

CSS Masters

CSS Masters Israel

CSS Front End Development Web Development UX Design UX