How to center things with style in CSS

freecodecamp.org/news/how-to-center-things-with-style-in-css-dc87b7542689/

June 17, 2018

by Stephen Sun

Sometime throughout our coding careers, we've all found ourselves frustrated with centering things in CSS (looking up how to center a div within a div on Google or Stack Overflow).

It can be one of the simplest tasks to do, but can quickly become confusing as you add more elements and styles into your page.

Since it's a pretty common problem, I've compiled a list of ways for centering in CSS into this guide. I've also included embeds/links with each example I created in CodePen. Feel free to fork, share, or copy as you wish!

View my CodePens here.

Without further ado, let's get into it!

Text-Align Method

The "text-align: center" method is perhaps the most common one you'll see for centering. It's used mostly for centering text on your HTML page, but it can be used to center divs as well.

The trick here is to:

1. Enclose the div that you want to center with a parent element (commonly known as a wrapper or container)
2. Set "text-align: center" to parent element
3. Then set the inside div to "display: inline-block"

In my example with the blue square, I enclose it with another div called "blue-square-container". In order to center my blue square, I had to create a parent element and set the display property of my blue square to inline-block.

This is because by default, a div's display property is set to block, meaning it will span the whole width of the page. By setting the display property of my blue square to inline-block, we ensure that the blue square will only span the width that I have set, which is 100px.

Adding multiple child elements within the parent element (blue squares in this example), will center all of them.

Margin Auto Method

Another common way of centering is using the margin auto method. Using this method, we don't need a parent element.

We can simply apply "margin: 0 auto" to our yellow box, as long as we have a defined width.

"margin: 0 auto" is shorthand for setting the top and bottom margins to zero, and the left and right margins to auto.

This is important, because without the 100px width I've defined, the browser will not be able to render the left and right margins needed to center the yellow box. By setting the width, the browser will automatically distribute the right amount of margin on either side of the yellow box.

The "0" portion can be set to any number of pixels you want for the top and bottom margins.

Another cool trick is just setting either margin-left to auto or margin-right to auto, which allows us to push our div to either the right or left side of the page, respectively (give this a try!).

Absolute Positioning Method

Absolute positioning an element allows us to essentially place the element wherever we want it on the page...with one drawback.

Absolute positioning removes the element from the flow of the page.

Why is this important?

Well it's important because this can cause overlapping of elements if used incorrectly.

If we we want to simply center an element horizontally on the page like we've been doing in the first two methods, there are three steps we need to remember:

1. Set the element's position property to absolute
2. Apply "left: 50%" to the element
3. Set a margin-left of half of the element's width

In this example, we use a green square (what a beautiful green!). It's the same size as the other examples, so our width is still 100px.

As you can see, I've given "position: absolute" and applied "left: 50%" to our green square. This tells the browser to move the left edge 50% to the right.

But if you're recreating this example, we don't want the left edge to be in the middle, we want the middle of the square to line up with the middle of the page.

This brings us to our last step. In order to line things up and offset the extra space, we apply a margin-left of half of the green square's width. In our case it's 50px (regardless of the width of the element, it will always be half).

Transform/Translate Method

Up until now we've only dealt with centering things horizontally, but what if we want to put something right in the middle of the page?

Let's center our red square both horizontally and vertically.

Though this method also uses absolute positioning and "left: 50%", I've also applied two more properties to the element.

By setting the top property to 50% as well, I'm telling the browser to line up the top edge of our red square with the middle of the page vertically. But like in the previous example, we don't want the edges to be lined up with the center, we want the center of our square to fit right on top of the center of our page.

This is where we apply a new property called transform.

There are many cool things you can do with transform, such as translating, rotating, and scaling animations, but for this example, we'll be using **translate.**

We give the transform property "transform: translate(-50%, -50%)" and voila!

Our red square is centered both horizontally and vertically!

I love this method, because regardless of what the width or the height of our element is, it will always be in the center of the page.

This method is used frequently in responsive design and doesn't require margins to be defined, like in the absolute positioning method.

Flexbox Method

If you're not familiar with Flexbox, that's okay! Flexbox is a layout module that provides a more efficient way of aligning and placing elements on the page.

If you're **interested in learning Flexbox (highly recommend)**, **Flexbox Froggy** is an awesome and fun way to learn (not affiliated of course — it's just what I used to learn Flexbox)!

The four steps to centering horizontally and vertically with Flexbox are the following:

1. HTML, body, and parent container need to have a height of 100%
2. Set display to flex on parent container
3. Set align-items to center on parent container
4. Set justify-content to center on parent container

Setting display to flex on the parent defines it as a flex container.

By setting align-items to center, we're saying that the children or flex items are to be centered vertically within the parent.

Justify-content works in the same way, but in the horizontal direction for our example.

I like using this method as well because again, it's both **responsive** and **doesn't require any margin calculations**.