

HTML & CSS

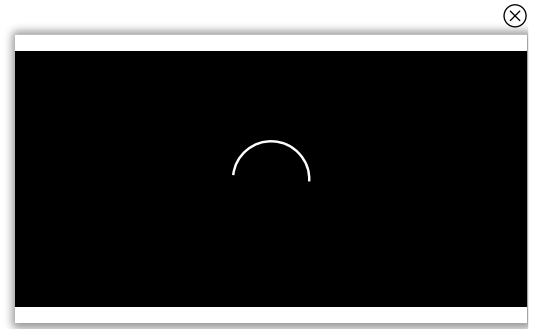
How to Center a Div Using

CSS



Fimber Elemuwa

September 29, 2022

Share     



In this article, we'll look at four ways to horizontally and vertically center a div using CSS Grid. Of course, these centering techniques can be used on any kind of element. We've previously covered how to center elements horizontally and vertically using [Flexbox](#) and [positioning with transforms](#).

Setting Up

Let's first create a container with a simple box element inside it that we'll use to demonstrate these centering methods. Here's the HTML:

```
<article>
  <div></div>
</article>
```

And here's our starting CSS:

 **50% off Annual Plans!** 700+ Assessments, Books and Courses! **05:53:02** [Get Now](#) 

```
article {
  width: 100%;
  min-height: 100vh;
  background: black;
  display: grid;
}

div {
  width: 200px;
  background: yellow;
  height: 100px;
}
```



In all our examples, we'll be using the `display: grid` property. The `<article>` element as a grid container and generates a block-level grid. (Here's [our demo template on CodePen](#) if you want to experiment with it.)

Now, let's look at the various ways to center our div.

1. Center a Div with CSS Grid and `place-self`

My favorite way to center an element with Grid is to use the `place-self` property. (You can read more about it [here](#).)

Centering our div is as simple as this:

```
article {  
  display: grid;  
}  
  
div {  
  place-self: center;  
}
```



The `place-self` property is a shorthand for the `align-self` (vertical) and `justify-self` (horizontal) properties (which are useful if you're just centering along one axis). You can experiment with them [in this CodePen demo](#).

Using `place-self` is so simple that it's an obvious go-to solution. But it's not the only way to center an element with Grid, so let's now look at some other methods.

An advantage of using `place-self` is that it's applied to the element itself, meaning that you can also use it to center other elements in the same container. (We'll add more div elements to the CodePen demo and see what happens.)



2. Center a Div with CSS Grid, justify-content and align-items

Let's now look at what's involved with using Grid with `justify-content` and `align-items` to center our div.

The `justify-content` property is used to align the container's items horizontally when the items don't use all the available space. There are many ways to set the `justify-content` property, but here we're just going to set it to `center`.

Just like the `justify-content` property, the `align-items` property is used to align the content in a container, but it aligns content vertically rather than horizontally.

Let's return to our test HTML and add the following code to the parent container:

```
article {
  display: grid;
  justify-content: center;
  align-items: center;
}
```



An apparent advantage of this method is that it involves less code, as the centering is handled by the container. But in some ways it's also a disadvantage to target the div via its container, as any other element in the container will also be affected.

3. Center a Div with CSS Grid and Auto Margins

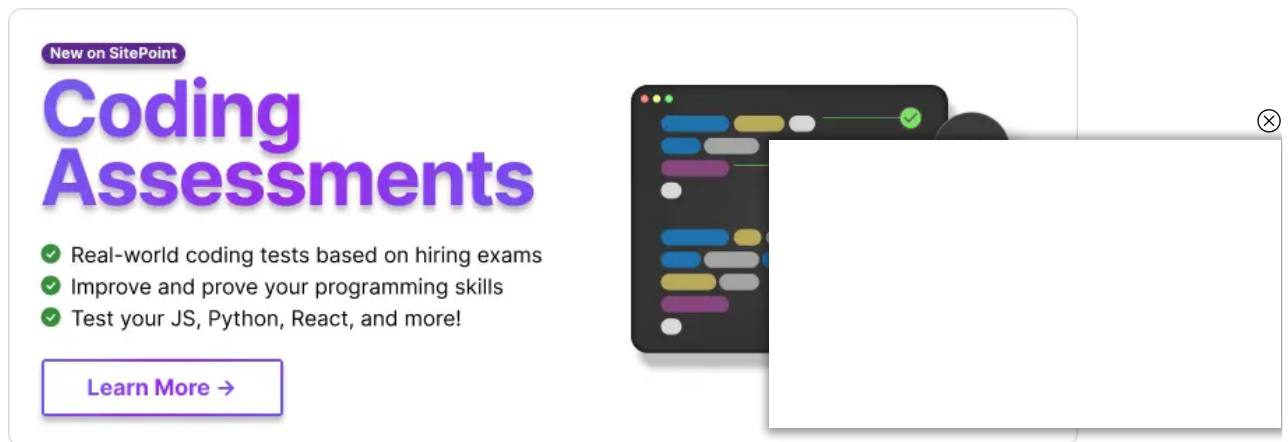
As always, we'll target the parent container with `display: grid`. We'll also assign the div an automatic margin using `margin: auto`. This makes the browser automatically calculate the available space surrounding the child div and divide it vertically and horizontally, placing the div in the middle:

```
article {
  display: grid;
}

div {
  margin: auto;
}
```



(As an aside, there are **lots of other cool things you can do** with CSS margins.)



4. Center a Div with CSS Grid and place-items

The `place-items` property is used to align items vertically and horizontally in a grid. It can be used to center our div by targeting the container like this:

```
article {  
  display: grid;  
  place-items: center;  
}
```



Like the `place-self` property, `place-items` is shorthand for two properties, in this case `justify-items` (horizontal) and `align-items` (vertical). You can experiment with them in [this CodePen demo](#).

In contrast to `place-self`, `place-items` is applied to the container, which gives it slightly less flexibility.

Conclusion

Each of these methods lets you center a div horizontally and vertically within a container. As I said, my preference is the the `place-self` method, mainly because it's applied to the

element being centered rather than the container. That's the same for the `margin: auto` method. But of course, if you're only looking to center your element in one direction, you can also use either `align-self` or `justify-self`.

In the demo examples, we've just used an empty div, but of course any element can be centered in a div and the centering will still work. And, once again, these centering techniques work on elements other than divs.

Further reading:

- [Flexbox or CSS Grid? How to Make Layout Decisions that Make Sense](#)
- [CSS Master, 3rd Edition](#)
- [An Introduction to CSS Grid Layouts](#)
- [Modern CSS Layout](#)

Share This Article



Fimber Elemuwa

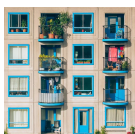
Fimber is a freelance frontend developer, Unity dev and technical writer who writes about the best programming practices, and likes making hard concepts easier. He's currently studying Microbiology at the University of Port Harcourt.



[centering css](#) [css grid](#) [css layout](#)



Up Next



5 Super CSS Grid Generators for Your Layouts

Maria Antonietta Perna

An Introduction to CSS Grid Layouts

Ahmad Ajmi



The CSS Grid Layout vs CSS Frameworks Debate

Maria Antonietta Perna

Building a Trello Layout with CSS Grid and Flexbox

Giulio Mainardi



How to Vertically Center Text and Icons in CSS

Guy Routledge

Introducing the CSS Grid Layout

Nitish Kumar

Stuff we do

- Premium
- Newsletters
- Forums

About

- Our story
- Terms of use
- Privacy policy
- Corporate memberships

Contact

- Contact us
- FAQ
- Publish your book with us
- Write an article for us
- Advertise

Connect



© 2000 – 2023 SitePoint Pty. Ltd.

This site is protected by reCAPTCHA and the Google [Privacy Policy](#) and [Terms of Service](#) apply.