Why doesn't "margin: auto" center an element vertically?

Asked 8 years, 11 months ago Modified 1 year, 10 months ago Viewed 21k times

58

As you can see in the demo below, `margin: auto;` centers the blue `div` horizontally, but not vertically. Why not?

```
.box {
  border: 1px solid red;
  width: 100px;
  height: 100px;
}
.center {
  background: blue;
  width: 50px;
  height: 50px;
  margin: auto;
}
```

```
<div class="box">
  <div class="center"></div>
</div>
```

Run code snippet Expand snippet

My question is not asking for workarounds.

html css margin

Share
Improve this question
Follow

edited Jan 27, 2016 at 21:45

asked Dec 31, 2015 at 22:30

Stickers
78.6k ● 24 ● 155 ● 194

It just doesn't. The use case for centering content horizontally was presumably much more common than doing so vertically. – TZHX Dec 31, 2015 at 22:36

1 @TZHX Technically, it *can* center vertically. See this example - jsfiddle.net/skdjhpo8 – Josh Crozier Dec 31, 2015 at 22:37

Vertical Alignment in CSS is a myth – Fabio Poloni Dec 31, 2015 at 22:40

2 @FabioPoloni Flexbox? – user3117575 Dec 31, 2015 at 22:41

@FabioPoloni ¯_(ツ)_/¯ Newer versions of IE do actually support flexbox, with certain caveats. – Dan Jan 1, 2016 at 14:40

5 Answers

Sorted by: Highest score (default)

39

As mentioned, this behavior is specified in section 10.6.2 of CSS2.1, and has remained unchanged from CSS2.

Block boxes are stacked vertically from top to bottom in normal flow. Furthermore, vertical margins may collapse, and only do so under certain circumstances (in your demo, the border on the parent element will prevent any margins on the child element from collapsing with its own). If you only have one such block box, and the height of the containing block is auto, then its top and bottom margins will be zero anyway. But if you have more than one block box in the same flow, or even out-of-flow boxes affecting the layout of in-flow boxes (in the case of clearance for example), how would you expect auto margins to resolve for those in-flow boxes?

This is why auto left and right margins are likewise zeroed out for inline elements (including atomic inlines) and floats (though horizontal margins never collapse). Inline-level boxes are laid along line boxes, and floats too obey unique layout rules.

Absolutely positioned boxes are a different story: since they are never aware of any other boxes in the same positioning context as themselves, auto top and bottom margins *can* be calculated for them with respect to their containing blocks without having to worry about any other boxes ever interfering.

Flexbox is also a different story: what sets flex layout apart from block layout is that flex items are by definition always aware of other flex items in the same flex formatting context, including the fact that there are none. In particular, neither can floats intrude into the flex container, nor can you float flex items to subvert this (although you can still remove a child element from flex layout completely with absolute positioning). Margins behave very differently with flex items due in part to this. See sections 4.2, 9.5 and 9.6.

Share Improve this answer Follow

edited Jul 26, 2017 at 4:14

answered Dec 31, 2015 at 22:58

BoltClock

722k ● 165 ● 1.4k ● 1.4k

37

Why...because the **W3C spec** says so.

> If 'margin-top', or 'margin-bottom' are 'auto', their used value is 0.

As to the actual "why"...the query should really be addressed there.

answered Dec 31, 2015 at 22:41

Paulie_D
115k • 13 • 160 • 182

24

It doesn't center the element vertically because it is a block-level element in the normal flow. Thus, the following rule applies:

> If `margin-top`, or `margin-bottom` are `auto`, their used value is 0.

It's also worth pointing out that the rule above also applies to the following elements as well: (see points 10.6.2 and 10.6.3 for more information and conditions).

- Inline replaced elements
- Block-level replaced elements in normal flow
- `inline-block` replaced elements in normal flow
- Floating replaced elements
- Block-level non-replaced elements in normal flow when `overflow` computes to `visible`

With that being said, absolutely positioned, non-replaced elements that don't have `top`, `height`, and `bottom` values of `auto` are an exception to this rule. The following applies from point 10.6.4:

> If none of the three `top`, `height`, and `bottom` are `auto` and if both `margin-top` and `margin-bottom` are `auto`, **solve the equation under the extra constraint that the two margins get equal values**.

See the example below demonstrating how an absolutely positioned element is vertically centered using `margin: auto`. It works because none of the three properties `top`, `height`, and `bottom` have a value of `auto`:

```
.box {
  border: 1px solid red;
  width: 100px;
  height: 100px;
  position: relative;
}
.center {
  background: blue;
  width: 50px;
  height: 50px;
  margin: auto;
  position: absolute;
  top: 0; right: 0;
```

```
  bottom: 0; left: 0;
}
```

```
<div class="box">
  <div class="center"></div>
</div>
```

Run code snippet Expand snippet

In addition, it's probably worth pointing out the following rule as well:

> If one of `margin-top` or `margin-bottom` is `auto`, solve the equation for that value. If the values are over-constrained, ignore the value for `bottom` and solve for that value.

This means that if the absolutely positioned element has a `margin-top` value of `auto` and a `margin-bottom` value of `0` (i.e., `margin: auto auto 0`), the element would be absolutely positioned at the bottom relative to the parent like in the example below:

```
.box {
  border: 1px solid red;
  width: 100px;
  height: 100px;
  position: relative;
}
.center {
  background: blue;
  width: 50px;
  height: 50px;
  margin: auto auto 0;
  position: absolute;
  top: 0; right: 0;
  bottom: 0; left: 0;
}
```

```
<div class="box">
  <div class="center"></div>
</div>
```

Run code snippet Expand snippet

Share Improve this answer Follow

edited Dec 31, 2015 at 23:25

answered Dec 31, 2015 at 23:02

Josh Crozier

241k • 56 • 400 • 313

17

> *Why doesn't `margin:auto` work vertically?*

Actually, it does – just not for every `display` value.

If `display` is `flex`, `margin: auto` centers both vertically and horizontally.

The same applies to `display: inline-flex`, `display: grid` and `display: inline-grid`.

```
.box {
  border: 1px solid red;
  width: 100px;
  height: 100px;
  display: flex; /* new */
}
.center {
  background: blue;
  width: 50px;
  height: 50px;
  margin: auto;
}
```

```
<div class="box">
  <div class="center"></div>
</div>
```

Run code snippet Expand snippet

Share Improve this answer Follow

edited May 23, 2018 at 13:13

answered Dec 31, 2015 at 23:36

Michael Benjamin

370k ● 110 ● 609 ● 725

3

It's because of the actual possibility of knowing the true height of the element in which you want to center vertically in. To understand that, first think about how auto horizontal centering works. You have a div which you've given it a width (fixed or percentage). The width can be calculated to certain degree. If it's fixed width, great. If it's flexible or responsive (percentage) at least you have a range that the width will cover before it hits the next breakpoint. You take that width, minus whatever it's inside and split the remainder on both sides.

Now, with that information, how could the browser calculate the infinite amount of variations in which your div will grow vertically? Keep in mind the size of the element,

wrapping of text, paddings, and responsiveness will also alter the width and force the text to wrap further, and on, and on it goes.

Is it an impossible task? Not really, has CSS spent time and effort covering this? Not worth their time, I guess.

And that is basically the answer I tell my students.

But....fret not! Bootstrap v4 alpha has figured out vertical centering!

EDIT

Sorry to edit this late but I thought you may want to consider this solutions to center vertically and it is pretty simple by making use of the calc function

```
<div class="foo"></div>

.foo {
  background-color: red;
  height: 6em;
  left: calc(50% - 3em);
  position: absolute;
  top: calc(50% - 3em);
  width: 6em;
}
```

See it **HERE**

Share Improve this answer

Follow

edited Jan 10, 2016 at 14:54

answered Dec 31, 2015 at 23:05

LOTUSMS

10.2k ● 18 ● 76 ● 150

5 It would be truer to say that **Flexbox** has figured our vertical centering and B4a has jumped on the bandwagon. – Paulie_D Dec 31, 2015 at 23:14

That, I wasn't aware of – LOTUSMS Dec 31, 2015 at 23:15