The `box-sizing` property can make building CSS layouts easier and a lot more intuitive. It's such a boon for developers that here at CSS-Tricks we observe International Box-Sizing Awareness Day in February.

But, how is it so helpful and beloved that it deserves its own internet holiday? Time for a little bit of CSS history.

Box Model History

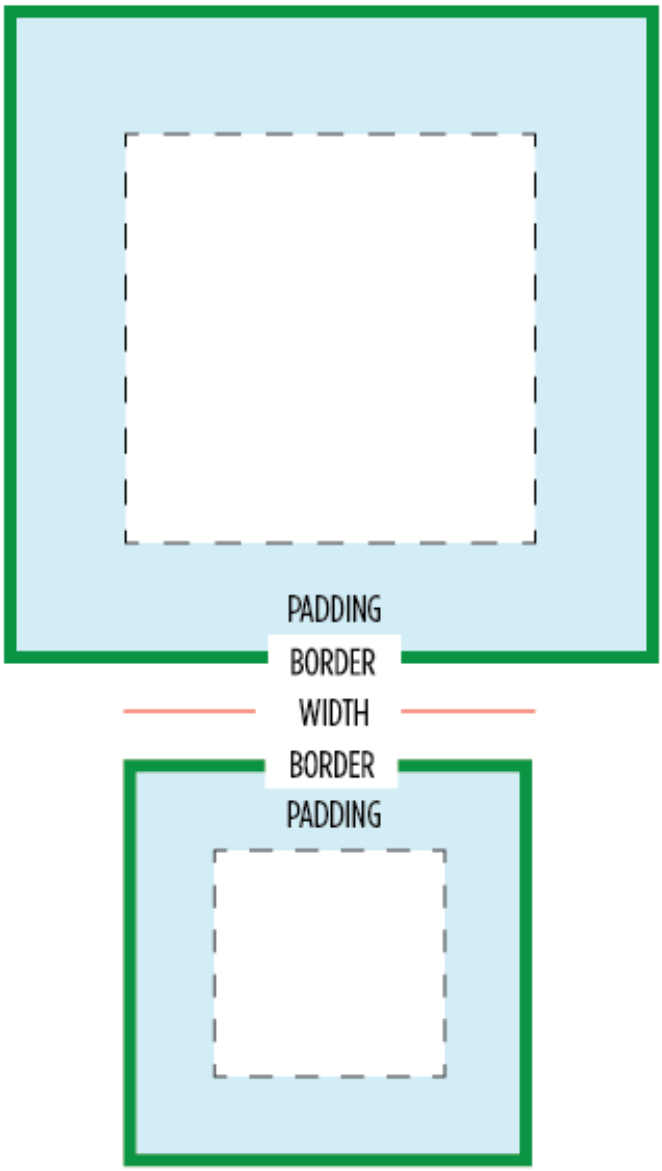
Since the dawn of CSS, the box model has worked like this by default:

width + padding + border = actual visible/rendered width of an element's box

height + padding + border = actual visible/rendered height of an element's box

This can be a little counter-intuitive, since the width and height you set for an element both go out the window as soon as you start adding padding and borders to the element.

Back in the old days of web design, early versions of Internet Explorer (<= IE6) handled the box model differently when it was in "quirks mode". The "quirks" box model worked like this: width = actual visible/rendered width of an element's box height = actual visible/rendered height of an element's box The border and padding values were moved inside the element's box, cutting into the width/height of the box rather than expanding it.



The box at the top shows the default box model. The box at the bottom shows what was once the "quirks mode" interpretation of the box model.

Some people preferred this "quirky" interpretation of the box model and considered it more intuitive. It's a valid point. Having the actual visible width of a box turn out differently from what you declared in the CSS is a bit mind bending.

But, in the days of fixed-width design, it wasn't particularly complicated to work with the default box model once you understood it. You could do a bit of arithmetic to figure out how many pixels you needed to trim off of an element's declared width or height to accommodate its padding and border. The problem for present-day developers is that those absolute pixel lengths don't translate to responsive design, so the same math doesn't apply anymore.

As responsive design (or, as it was once known, "fluid" or "liquid" layout) started to gain popularity, developers and designers wished for an update to the box model. The great designer Jon Hicks, known for his excellent fluid width

designs, had this to say on the subject in the CSS Wishlist we put together in 2008:

> *I would love a different box model! I find it bizarre that padding and border add the width of an object, and would love to be able to give something like a textarea 100% width and 3px padding without worrying what it's going to do the layout. Perhaps something like padding-inside as a new selector?*
>
> *In that vein I also wish I could specify a 100% width for an element, minus a set fixed width. Again, very useful when creating fluid designs with form elements!*

Present-Day `box-sizing`

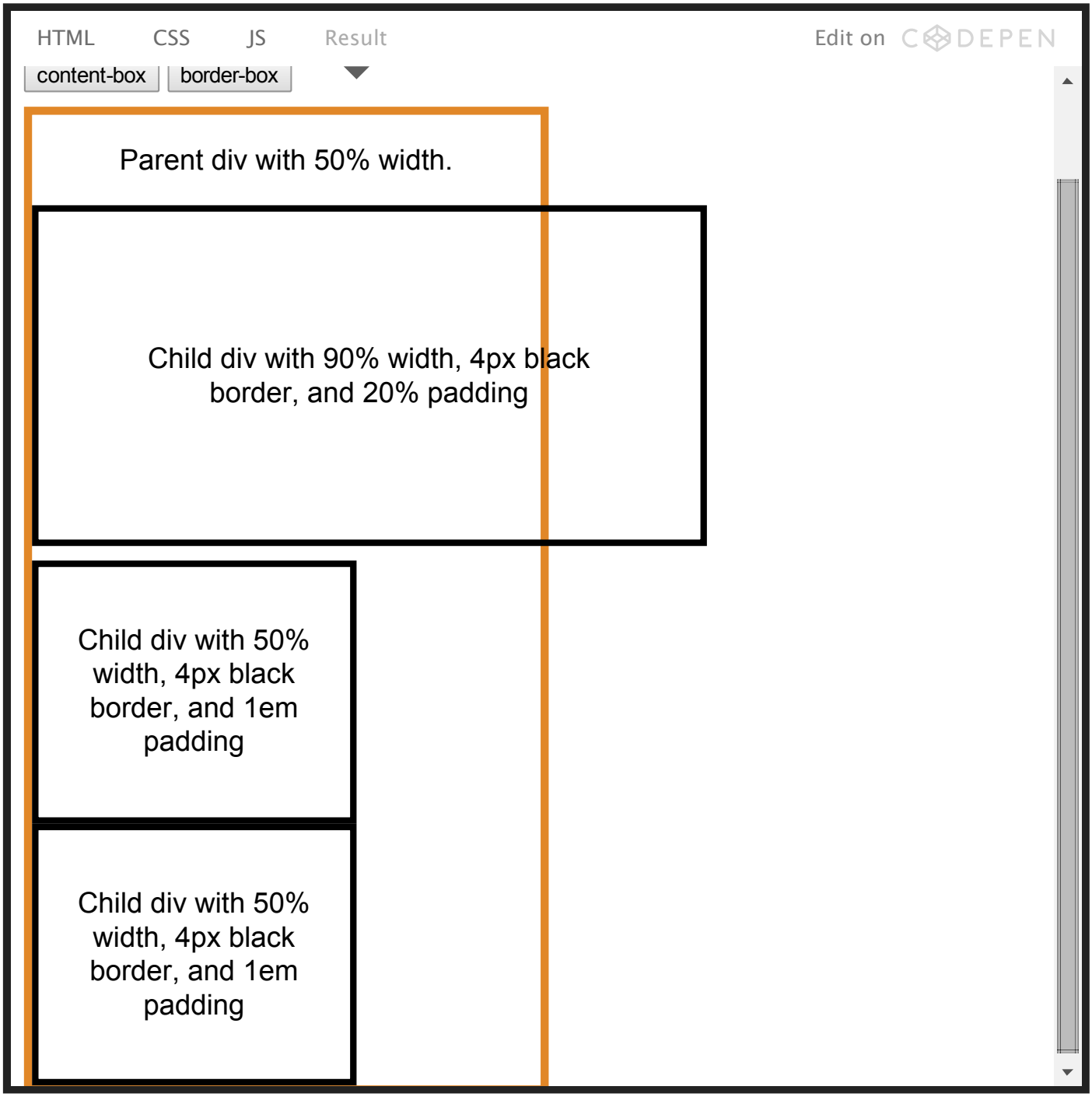
Those wishes were granted when the `box-sizing` property was introduced in CSS3. Though `box-sizing` has three possible values (`content-box` , `padding-box` , and `border-box`), the most popular value is `border-box` .

Today, the current versions of all browsers use the original "width or height + padding + border = actual width or height" box model. With `box-sizing: border-box;` , we can change the box model to what was once the "quirky" way, where an element's specified width and height aren't affected by padding or borders. This has proven so useful in responsive design that it's found its way into reset styles.

At this point you may be asking yourself, "Is it possible that Old IE did something right?" Plenty of people think so.

Demo

This demo shows how `border-box` can help make responsive layouts more manageable. The parent `div` 's width is 50%, and it has 3 children with different widths, padding, and margins. Click the `border-box` button to get all the children in the right place inside the parent.



Good, Better, and (Probably) Best `box-sizing` Reset Methods

The "Old" `border-box` Reset

The earliest `box-sizing: border-box;` reset looked like this:

CSS

```
* {
  box-sizing: border-box;
}
```

This works fairly well, but it leaves out pseudo elements, which can lead to some unexpected results. A revised reset that covers pseudo elements quickly emerged:

Universal Box Sizing

CSS

```
*, *:before, *:after {
  box-sizing: border-box;
}
```

This method selected pseudo elements as well, improving the normalizing effect of `border-box`. But, the `*` selector makes it difficult for developers to use `content-box` or `padding-box` elsewhere in the CSS. Which brings us to the current frontrunner for best practice:

Universal Box Sizing with Inheritance

CSS

```
html {
  box-sizing: border-box;
}
*, *:before, *:after {
  box-sizing: inherit;
}
```

This reset gives you more flexibility than its predecessors — you can use `content-box` or `padding-box` (where supported) at will, without worrying about a universal selector overriding your CSS. We went into more depth on this technique and the reasoning behind it in "Inheriting `box-sizing` Probably Slightly Better Best Practice". One potential gripe with it is that `box-sizing` isn't normally inherited, so it's specialized behavior, not quite the same as something you'd normally put in a reset.

Vendor Prefixes

Every current browser supports `box-sizing: border-box;` unprefixed, so the need for vendor prefixes is fading. But, if you need to support older versions of Safari (< 5.1), Chrome (< 10), and Firefox (< 29), you should include the prefixes `-webkit` and `-moz`, like this:

CSS

```
html {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
*, *:before, *:after {
  -webkit-box-sizing: inherit;
  -moz-box-sizing: inherit;
  box-sizing: inherit;
  }
```

Known Issues

`box-sizing: border-box;` is supported in the current versions of all major browsers. The less-commonly used `padding-box` is only supported in Firefox at the moment. There's more comprehensive information about browser support in our `box-sizing` almanac entry.

There are a few issues with older versions of Internet Explorer (8 and older). IE 8 doesn't recognize `border-box` on elements with `min/max-width` or `min/max-height` (this used to affect Firefox too, but it was fixed in 2012). IE 7 and below do not recognize `box-sizing` at all, but there's a polyfill that can help.