written by Steve Rydz on 2 September 2015 in engineering, front-end

thinking outside the box (model)

The CSS box model has always been a pain to work with, until a couple of years ago when the box-sizing property started landing in browsers. Finally we could add padding and borders to components without completely breaking the layout of our pages.

This property was so well received that many developers began adding it to their boilerplates by default, typically as follows:

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

This ensures that every element on the page `border-box`, however there is one catch. What if you need to change the box-sizing property, for example if you were using a third-party component that had been styled using the default `content-box` model? Surely you just reset that property on that component and all will be fine?

```
.third-party-component { box-model: content-box; }
```

Doing this will reset the `third-party-component` div to `content-box`, but any elements inside it will still be using `border-box`. So how do we fix this? We could just set the `box-sizing` property on every element likely to be nested within that component, but that could get messy very quickly.

A better solution is to avoid this problem in the first place by setting the `box-sizing` property as follows:

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

This way, each element will inherit the `box-sizing` property of its parent, which means by default any element will have the `box-sizing` value of `border-box`, but if it's reset to `content-box` at any point, such as in our example above, any children will also use the same value.

If you enjoyed the read, drop us a comment below or share the article, follow us on Twitter or subscribe to our #MetaBeers newsletter. Before you go, grab a PDF of the article, and let us know if it's time we worked together.