

# The problem with the EM unit.

The EM unit can cause unintended results when not understood. Let me save you the stress.

Let's go straight to an example:

## Example

Consider the markup below:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8" />
5 <title>A Simple Page</title>
6 <link rel="stylesheet" href="s
7 </head>
8 <body>
9   <div class="one">
10     One Hello World!
11     <div class="two">
12       Two Hello World
13       <div class="th
14       Three Hello Wo
15     </div>
16   </div>
17 </div>
18 </body>
19 </html>
20
```



Please look very closely. Class, `.one` is a parent to `.two` and `.two` is a parent to `three`. Some deep nested stuff going on there.

So, if you styled the text in `.one` like this:

```
1 .one {
```

```
2 | font-size: 1.5em  
3 | }
```



What would be the value of the font size in pixels? Yes, 24px. Remember, 1em is equal to 16px, the browser's default size. 1.5em is equal to  $1.5 * 16\text{px}$ .

Alright, that's done.

In the nested class `.two`, what happens if you did this:

```
.two {  
  font-size: 1.5em  
}
```



What would the font-size be? 24px?

No! Sadly, no!

In case you missed it, I said this in the last section:

1em may also represent the font size of the parent element. Which is why it is loosely described as the “current font size”.

In `.two` the font size would be  $1.5 * 24\text{px}$ , i.e 36px

Where did the 24px come from? The parent element, `.one`

It's the same with the nested class, `.three`

```
.three {  
  font-size: 1.5em  
}
```



This wouldn't be  $1.5 * 16\text{px}$ . It'd be 1.5 times the font-size of the parent element. i.e  $1.5 * 36\text{px}$ .

This is probably NOT the behaviour you expect. In many cases you just want a simple metric to scale against.

Why the hassle?

In the next lesson, I'll introduce you to the `rem` unit. It pretty much resolves

this weird (maybe, not weird) issue with the `em` unit.