# 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:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>A Simple Page</title>
<link rel="stylesheet" href="styles.css" />
</head>
<body>
  <div class="one">
            One Hello World!
            <div class="two">
                Two Hello World
                <div class="three">
                Three Hello World
                </div>
            </div>
  </div>
</body>
</html>
```

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:

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

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 * 16px.

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 * 24px, 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 * 16px. It'd be 1.5 times the font-size of the parent element. i.e 1.5 * 36px.

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.