[Articles](http://fvsch.com/articles/)[Code](http://fvsch.com/code/)[Photos](http://fvsch.com/photos/)

***Alternatives to CSS’s em unit***

March 2016

I’ve been coding CSS for 10 years now, and in those 10 years the world of CSS units can be summed up as:

1. Everybody uses px most — if not all — of the time.
2. CSS experts encourage everyone to use em for text, then (in the past few years) in more and more places: for element sizing, padding, in media queries etc.

Of course the history and present of CSS units is much more rich. I still remember when Google Docs used pt and it messed up the rendering in Firefox on Linux because it was the only configuration that actually tried to show correctly-sided typographic *points* (in accordance with CSS2, but CSS3 fixed this issue by saying that pt is definitely not a point, just roughly 1.333333333px so please don’t bother thank you very much).

Anyway, in the good old days before responsive design, everyone usedpx for everything, including for font-size because the Photoshop mockup said Arial 11px and when people tried to use em as recommended by experts… well you would get some calculation or nesting wrong and end up with 10.7888888px or 12.33333333px and browsers would round it up or down and the client would highlight***text should be 11px Arial here*** in bold red italic in paragraph 18 of their feedback email.

The em unit is notoriously hard to work with, mostly because:

* 1em can mean 3 different things, depending on where you use it.
* Nesting elements which all have font-size: Xem declarations can wreak havoc in a web page.
* The spirit of “I want this text to be 20% bigger than text in its container” was cool but hard to reconcile with browsers that rounded font-sizes to pixels anyway (so we had to use verbose code, em calculators on websites, mixins and whatnot) and clients that wanted the exact same pixel value as in the Photoshop comp.

That unit can still do a few things right, including font-sizeing, but it’s nice that we have other choices nowadays. As Ben Schwartz [wrote recently](https://twitter.com/benschwarz/status/710790924943527936):

*“Use ems” is not the answer to CSS sizing units. em, rem, vh, vware all options in different contexts — learn them.*

Let me add old trusty px to the list. That unit got a bad rep, but except for one specific thing it can be a perfectly decent choice.

WHAT EM MAPS TO

Quick reminder so that we’re on the same page regarding the em unit. It has many meanings, and frankly that’s a design flaw in CSS, because it makes that unit a bit magic and hard to learn, but here we are andem can mean:

1. With all CSS properties that accept em values except one, 1emmeans “calculated font-size of the element”. That meaning is alright.
2. In a font-size declaration, 1em means “calculated font-size of the parent element”. This special-case meaning should have been avoided (IMO, and hindsight’s 20/20 etc.) — especially since percent values for font-size do the exact same thing.
3. In a media query, 1em means “default font-size as set up by the browser and the user”. The most common value is 16px but [it definitely can be different (smaller or bigger)](https://nicolas-hoizey.com/2016/03/people-don-t-change-the-default-16px-font-size-in-their-browser.html). This is not so bad but still adds to the confusion.

Meanwhile, the rem unit in media queries also maps to “default font-size as set up by the browser and the user”, which is *different from the root em value that rem usually maps to* (yay for another design flaw),[except in Safari which is buggy](http://zellwk.com/blog/media-query-units/). Don’t use rem in media queries, ever.

In my wildest dreams, font-size would not accept em values (percentage values are enough, and less confusing), media queries would not accept em values and would accept rem values without changing the meaning of rem in that context. Well, too late.

ALTERNATIVE FOR FONT SIZES: REM, %, VW…

Even though em are a pain to work with, it’s still valuable to try to adapt to the device’s or user’s *default font size*, which is not always16px. Maybe it’s a lost war at this point, with Bootstrap setting the root font-size in pixels and most websites not managing this well — what user is going to change their browser settings if it has no effect on 90% of websites? — but it’s still a good practice to follow *if it can be managed reasonably*.

*Font sizes with rem*

Your main tool for that is the rem unit. It maps to the computed value of font-size on the root element (usually the page’s html element).

Let’s say that Lisa is currently using a desktop browser and has not changed her system or browser settings for websites’ text size. By default, she gets 16px text. Meanwhile, Alok has configured his browser to use “big” text, which happens to map to a 24px default.

body {

font-size: 1.25rem; /\* Lisa gets 20px, Alok gets 30px \*/

}

h1 {

font-size: 2rem; /\* Lisa gets 32px, Alok gets 48px \*/

}

Of course you will need to manage text wrapping in your CSS in almost all elements of the design, but if you’re writing robust CSS that’s already a given. Test everything with text that wraps to 2 or 3 lines. Learn a few techniques to manage that. It’s not hard or time consuming once you’re used to it.

I recommend not changing the font-size of the html element, so that you can use rem more easily. For instance if you don’t change the base font-size value, then “I want this title to be 50% bigger than the user’s prefered text size” becomes: h2 {font-size: 1.5rem}. Easy.

Another technique is to make 1rem map to 10px or 20px *for a majority of users*:

html {

font-size: 125%; /\* Lisa: 1rem=20px, Alok: 1rem=30px \*/

}

body {

font-size: 1rem; /\* Lisa gets 20px, Alok gets 30px \*/

}

h1 {

font-size: 1.6rem; /\* Lisa gets 32px, Alok gets 48px \*/

}

or:

html {

font-size: 62.5%; /\* Lisa: 1rem=10px, Alok: 1rem=15px \*/

}

body {

font-size: 2rem; /\* Lisa gets 20px, Alok gets 30px \*/

}

h1 {

font-size: 3.2rem; /\* Lisa gets 32px, Alok gets 48px \*/

}

*Font sizes with percentages*

Now that’s not really an alternative to em since it does the same thing, but it avoids the confusing meanings of em so maybe try it. :)

I feel that em or % can be used effectively if you use a component-based approach to CSS, and your component is unlikely to have too many nested components inside it.

.MyWidget {

font-size: 1rem;

}

.MyWidget-title {

font-size: 140%;

}

.MyWidget-button {

font-size: 110%;

}

.MyWidget-smallPrint {

font-size: 85%;

}

Here the elements in the MyWidget components have text sizes relative to their parent/ancestor, and since the base MyWidget font size is inrem there’s little risk that it would change depending on where we put the widget in the DOM. It also becomes easy to provide this component in two “sizes”, normal and small:

.MyWidget--small {

font-size: .85rem;

}

<!-- Normal style -->

<div class="MyWidget">

<h2 class="MyWidget-title">…</h2>

…

</div>

<!-- Smaller style -->

<div class="MyWidget MyWidget--small">

<h2 class="MyWidget-title">…</h2>

…

</div>

*Font sizes with a dash of vw*

The vw and vh units map to 1 percent of the viewport width or height. They can be used to adapt some text to the viewport.

.mySpecialText {

font-size: 3vw;

}

The problem here is that it can make the text really big or really small depending on the viewport. For instance, 3vw on a 320px-wide viewport is 9.6px, which is really small.

You’re better off starting with a minimum value that is big enough, and add a little bit of viewport width and/or height.

.mySpecialText {

font-size: calc(1rem + .25vw + .25vh);

}

That’s a fun trick, but make sure to test it on both small and very big screens.

ALTERNATIVE FOR MEDIA QUERIES: PX

Now that’s a fun one.

*A brief history of em in Media Queries*

When responsive design got its big debut, most code examples used pixels in media queries: @media (max-width:1024px) {…}

Then the CSI: CSS folks crashed the party, and after a few drinks they were like “you see my friend em is a best practice so you should respect the user preference and use em in media queries, alrite??”

It was a bit strange because the user preference they were talking about was for *text size* and they were arguing that the whole *page layout* should follow the *text-size*.

Imagine that you want to buy a cool hardcover book, but when skimming through it in the bookstore you find that the text is really small and hard to read. On the cover there’s a big sticker that says “Bigger text? Ask for the special edition.” So you go to a store clerk, ask about the big text edition, and they hand over an enormous book. The text is indeed bigger, but it’s the same layout, with pages twice as large and twice as tall. The thing doesn’t fit in your bag, it’s heavy, and most importantly *your eyes get tired when reading because of very large — physically large — line length*.

Now if you consider that several web browser have two user settings,

1. default text size and
2. default zoom level,

perhaps *respecting the user’s preference* means not making the whole site zoom depending on the text size, but just keeping both separate.

Anyway, using em in media queries got popular because until early 2014 some browsers (well, WebKit) had a bug where zooming the page triggered the em-based media queries, but not the px-based ones. This has been fixed for a while, so using px in media queries is fine.

*Don’t use rem in Media Queries*

I wrote about it earlier, but let me remind you to not use rem in Media Queries, ever. Two reasons:

1. It means the same as em in this context, and *is not the same as remeverywhere else* which is deeply WTF-inducing.
2. Turns out it’s buggy in Safari anyway.

*Do use em, but be careful*

If you do use em-based media queries, you must not use pixel values for container widths (or min-width, or even max-width). Because if you’re doing that, you’re probably converting your media queries from pixel values to “that pixel value divided by 16”, which will work for Lisa but not for Alok. Alok may end up with a broken layout.

For details, see [possible issues with em-based media queries](http://fvsch-com.main.dev/code/em-media-queries/).

Also, remember to avoid min-width and max-width media queries with the same value:

@media (min-width: 40em) {}

/\* If screen is exactly 40em, this will override the first media query \*/

@media (max-width: 40em) {}

/\* But this one is safe \*/

@media (min-width: 39.99em) {}

*Do use px in media queries*

If your container widths (or max-widths or min-widths) or set in pixels then you should use px in your media queries. It’s fine, really. It’s perfectly fine.

Also the same issue with overlapping min-/max- queries applies:

@media (min-width: 750px) {}

/\* If screen is exactly 750px, this will override the first media query \*/

@media (max-width: 750px) {}

/\* But this one is safe \*/

@media (min-width: 759px) {}

On the other hand, if you have set your container dimensions in em orrem (I recommend rem here), then you should use em-based media queries.

Bottom line: don’t mix and match *layout* dimensions. Either go full pixels (for the main layout and media queries, not for text and not necessarily for padding etc.) or full text-relative (em in media queries,rem or em for block widths).

WHAT EM IS GREAT FOR

Remember when we listed three different meanings of em?

*With all CSS properties that accept em values except one, 1emmeans “calculated font-size of the element”.*

This one works quite well for ***text-level spacing***:

/\* A "Button component" example, really?

They’re like OOP examples with a "Car" class. \*/

.Button {

display: inline-block;

vertical-align: middle;

padding: .5em 1em;

font-size: 1rem;

}

.Button--big {

font-size: 1.5rem;

}

If we set the padding in pixels, it won’t change with the default text size. If we set it in rem, it will change, but when we apply the bigger text variant (Button--big), the padding will keep its relatively small size. Finally, if we use em, the padding will follow the font size.

With rem instead of em, the example above would require more code:

.Button {

display: inline-block;

vertical-align: middle;

padding: .5rem 1rem;

font-size: 1rem;

}

.Button--big {

padding: .75rem 1.5rem;

font-size: 1.5rem;

}

Also consider using em for margins between paragraphs, titles, etc. (That’s what default browser styles do, so no need to break that.)

h1, h2, h3, h4, h5, h6 {

margin-top: 1.5em;

margin-bottom: .75em;

}

There are a lot of cool uses for em, which I won’t delve in here, as this article is already long enough.

Oh and by the way, have you ever used the ch unit?