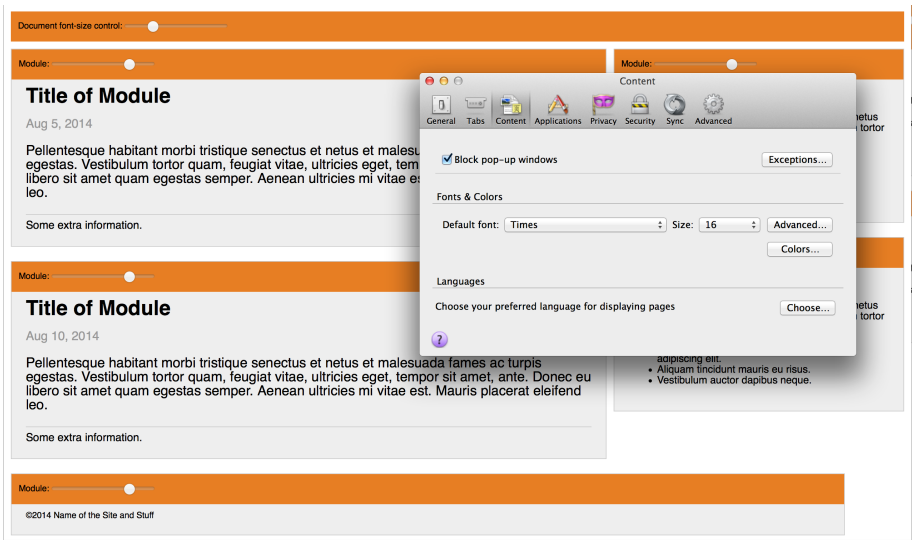
USING PIXELS IS NOT VERY POLITE

A few days ago **Chris Coyier** wrote an article about [which font-size units to use](#) *(http://css-tricks.com/rems-ems/)*. It's a very clever idea. He basically says to use the `em` for font-sizes of elements like `<h1>`, and use `rem` to adjust the font-size of different components on the page. This definitely makes sense. It sounds like a similar solution to the one I describe [here](#) *(https://vasilis.nl/nerd/dislike-rem-unit/)*. There is one important flaw in this article though. Chris defines the root font-size in pixels. This is an accessibility problem.

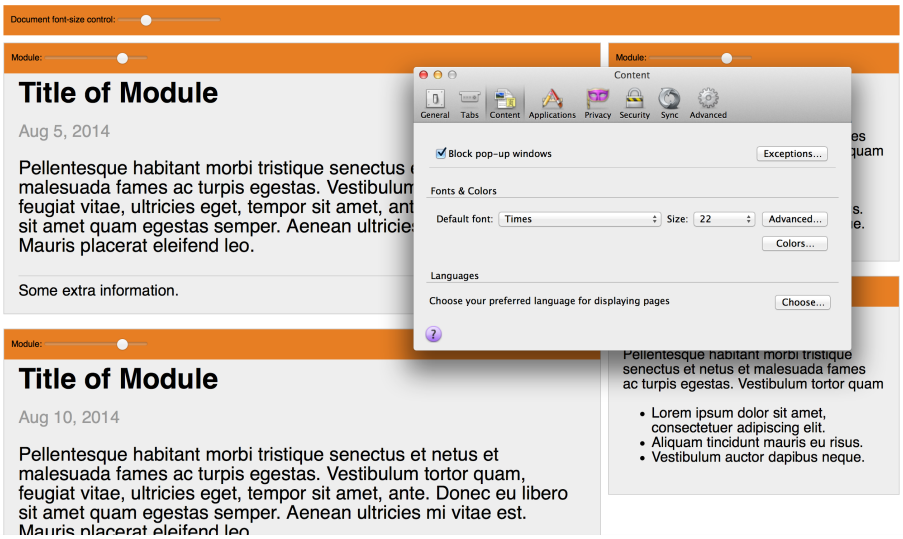
CUSTOM FONT-SIZES

If I need a bigger font-size in my browser I can easily do that by adjusting a setting. You would assume that all fonts would show up bigger now, but unfortunately, *fonts defined in pixels do not change to the users' prefs*. Let me illustrate. Here's a screenshot of the little tool Chris created to play around with his idea.



(http://ndoe.nl/screen/screenshot-1397114420.png)

This is how things were designed, and everything looks fine. Now in the next screenshot you can see what happens if you make the font-size bigger in your settings. All font-sizes that are defined in ems and rems have grown. But if you look at the labels in the tools you'll see that they haven't changed. These font-sizes are defined by the *root* font-size, which is defined in pixels. This is unreadable for people who need this preference.

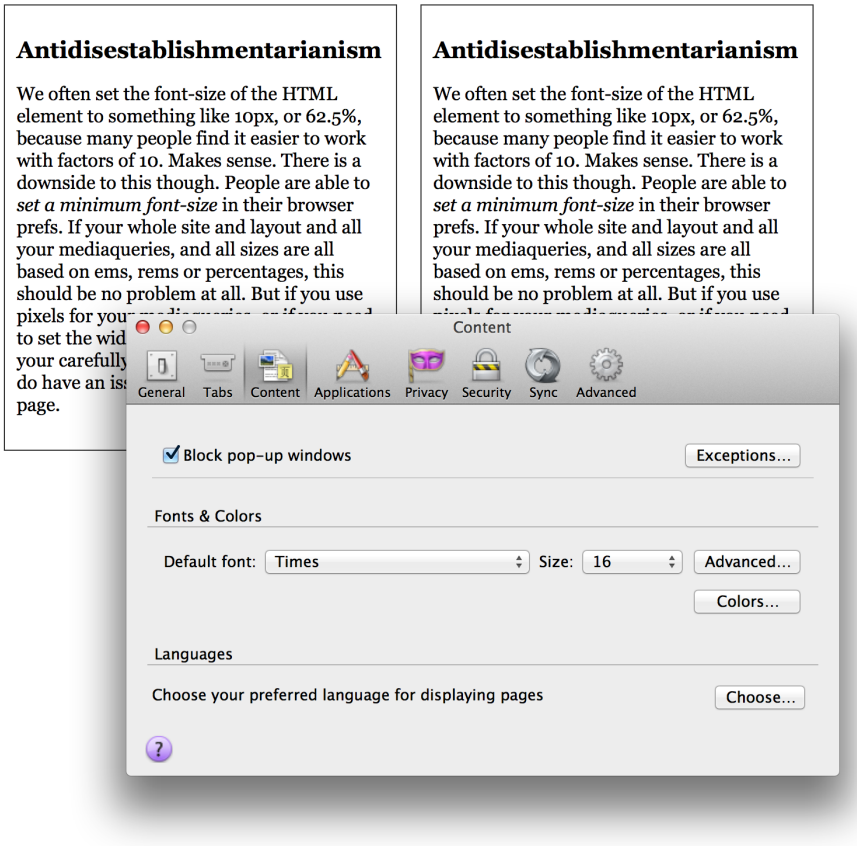


(http://ndoe.nl/screen/screenshot-1397114444.png)

It's very easy to prevent this issue. Just define the root font-size in `em` or percentages.

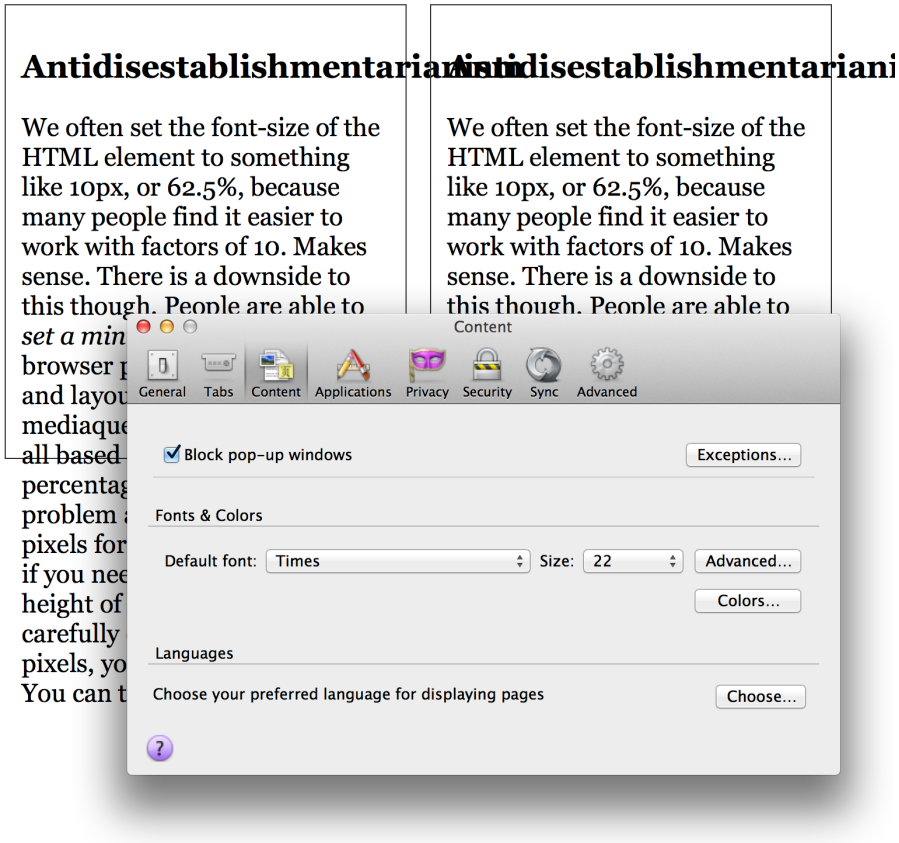
MEDIA QUERIES

There's another issue with Chris' example. He uses pixels for his media-queries. You should use `em` as a unit there too. And don't use pixels. Here's an example which uses `em`s for font-sizes and pixels for mediaqueries and widths. With default setting it looks like this:



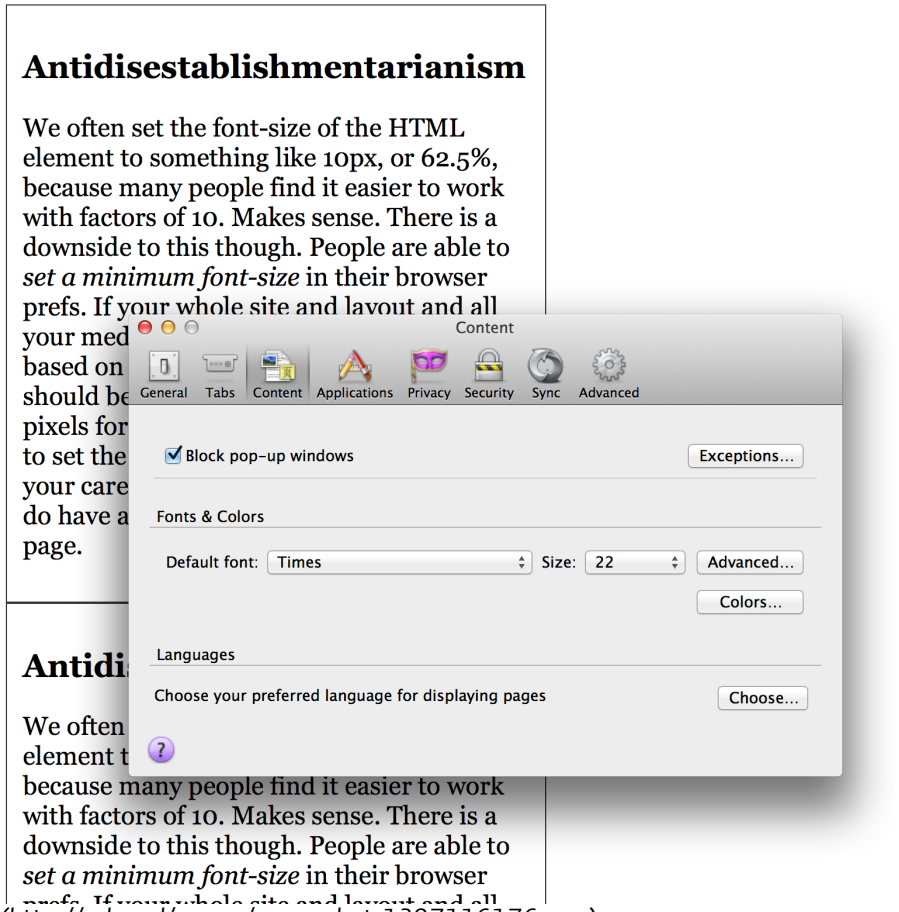
(http://ndoe.nl/screen/screenshot-1397115920.png)

Everything looks fine. But when we increase the font-size in our settings we see that things break terribly:



(http://ndoe.nl/screen/screenshot-1397115980.png)

[Here's the example for you to play with](https://vasilis.nl/nerd/code/rem/10px.html) *(https://vasilis.nl/nerd/code/rem/10px.html)*.
Here's a screenshot of the same example, with increased font-sizes, but this time with widths and mediaqueries based on ems.



(http://ndoe.nl/screen/screenshot-1397116176.png)

As you can see, the mediaqueries and the widths now react to the font-size in the users' settings, and not to pixels. This results in a layout that scales proportionally and doesn't break. [You can try it yourself here](https://vasilis.nl/nerd/code/rem/1em.html) (*https://vasilis.nl/nerd/code/rem/1em.html*).

NO PIXELS!

So again. If you want to respect the users' needs, and if you want to create truly fluid web stuff, don't use pixels for font-sizes, measurements and mediaqueries. And you should definitely use Chris' method, but be sure to change all the pixel-values to scalable `em`s.

WRITTEN BY VASILIS VAN GEMERT ON 10-04-2014

RELATED POSTS:

1. [Why I dislike the rem unit](https://vasilis.nl/nerd/dislike-rem-unit/) (*https://vasilis.nl/nerd/dislike-rem-unit/*)
2. [Use the em and the rem for the right use cases](https://vasilis.nl/nerd/use-em-rem-right-use-cases/) (*https://vasilis.nl/nerd/use-em-rem-right-use-cases/*)
3. [You should understand CSS units](https://vasilis.nl/nerd/you-should-understand-css-units/) (*https://vasilis.nl/nerd/you-should-understand-css-units/*)
4. [The weird 62.5% antipattern](https://vasilis.nl/nerd/weird-62-5-antipattern/) (*https://vasilis.nl/nerd/weird-62-5-antipattern/*)
5. [Text zoom options on websites, do you need them?](https://vasilis.nl/nerd/text-zoom-options-websites-need/) (*https://vasilis.nl/nerd/text-zoom-options-websites-need/*)

COMMENTS

MARTIJN SALY REPLIED ON 10/04/2014 #

Using ems or any other scalable unit for everything will eventually cause rounding problems for low-DPI screens (nearly all desktops and most laptops).

If you keep in mind what each value translates to, in pixels, for normal DPI, for normal font size pref, things should still look fine for most visitors in most situations on most devices.

VASILIS REPLIED ON 10/04/2014 #

Hi Martijn, thanks for your comment. I'm not sure I understand what you're talking about when you talk about rounding errors. Do you have an example of what you mean?
And about the other remark: Sure, using pixels will look good for most people. But that's not the point. I'm talking about a website being usable/unusable for people who need an adjustment. The thing I like about the web is that we can make stuff that's accessible to many more than just 'most visitors'. I'm aiming higher (-:

DAVID HUND REPLIED ON 10/04/2014 #

Good points Vasilis. I can see where Chris is coming from now that front-

end development focusses more and more on creating self-contained, reusable & modular, *components*.

A component must be *robust* in that it should be able to be implemented in a wide variety of contexts. E.g. it should not break when used in another component that messes with `font-size`.

I've found very little issues with a couple of simple guidelines:

Figure out a value for the **Base Font Size** for the content: mostly the body-text. 16+ px seems a good start.
Set the the value of this **Base Font Size** as *percentage* on the `html` root element: `html { font-size: 100%; /* ~= 16px */ }`
Define as ***little font-sizes as possible***. In other words: only define *exceptions* to the **Base Font Size**
When you need to explicitly *override* the **Base Font Size**, do so in `em`s

That's it, basically. There will be no need to define a `font-size` for `<ul>` or `<li>` and consequently there are no issues with *compounding* `em` values, etc.

The main elements you would explicitely start defining font-sizes for are headings.

As a side note I keep to the following, related, guidelines which also help:

Use percentages (%) for layout styles (`width`)
Use `rem` for `padding` / `margin`: where these are often related more to the Base Font Size, than to the component's font-size.
Try to keep vertical margins to One Direction (bottom) only
Define all Media Queries in `em`'s (and remember that these will be calculated to an em-value of ~16px, *not* ***your*** Base Font Size!)

Maybe I'm missing something and this is being way to simplistic, but I've found very little issues with it.

Unless maybe you need to have *very* different font-sizes between components in one site. But why would you need that? Also: soon we might have Scoped Styles ;)

VASILIS REPLIED ON 10/04/2014

Thanks for the fantastic comment/blog post, David! Those guidelines you write down are exactly the guidelines I use, except for margins since *I actually like the weird collapsing margin behaviour*. But I understand that normal people hate it (-:
I was planning on writing a blog post about using (font-)sizes and units, but you did a better job. Thanks!

MARTIJN SALY REPLIED ON 10/04/2014

Sorry that was a quick thing I typed on my phone. Lemme explain. When using ems, one way or another, the browser is going to have to round measurements off to pixels, because it renders a page in pixels in the end. Some browsers do sub-pixel rendering, but that will make things look blurry sometimes.

For example, say you declare font-size:125% on the and 60% on the . That'll get you a nice base font size of 12px in normal cases. For folks using more

dpi or a bigger font, it means more than 12px. Now you start usings ems for measurements of elements. Say a button needs to be the equavalent of 30px in size. So you make it 2,5em. Good. But what if it needed to be the equivalent of 40px? You could give it an odd size of 3.33333em, which *should* be okay in theory.

Now what if you decide to use tenths of ems only? Fine. Font size is the equiv of 12px, right? So A header might be 1.3em. That's 15.6px. The browser might render this as 15.6px text, or as 16px. Who knows. Now, there's another thing inside of that header that needs to be, say, the equiv of 10px. Now it gets hairy. First, let's define those 10px as 0.641025641025641025641025641025 64em. Ehh, no :) Let's make that 0.6em, as previously decided. That would become 9.36px.

Above uses two perfectly nice lengths: 1.3em and 0.6em. They look nice and friendly and "rounded", but will produce artefacts in rendering because the browser has no way of knows where to position such elements. Does it go on the pixel on the left or the one on the right? This is how I've had two elements, mathmatically seamlessly pressed together, with a seam in between. Sometimes. Why? Because rounding to whole pixels. No problems on mobile though, because HDPI.

I don't currently have any examples at hand to show you what I mean, because it already solved these problems in my projects :)

Anyway, my conclusion is to just be very wary of pixel rounding, at look out for situations where rounding errors may produce visual artefacts. If an element is positioned off by a pixel or two, I don't care. We're passed pixel-perfection. But If there exists a seam between two elements where it shouldn't be, or when an element is ever-so-slightly blurry, you've got a rounding problem.

VASILIS REPLIED ON 10/04/2014 #

Thanks for your long explanation, Martijn. I think I understand the problem. To be honest, I think it's mainly caused by trying to translate pixels to a web-friendly unit, and not by ems. If we forget about pixels and simply start thinking in ems, we probably won't have these issues, because we won't have to do these crazy calculations.
But I might be wrong here. If you ever find a good example of a rounding error, please do let me know!

MARTIJN SALY REPLIED ON 10/04/2014 #

Another slightly-related problem is the root font-size. Just any font-size won't do. Here's why.

In the past, I've always set the HTML font-size to 62.5%, because that makes for a comfortable 10px base font-size. However, in IE9-11 things got ever-so-slightly blurry. Especially once I discovered it was a good idea to size images using ems too. A 200×200 image *should* be 20x20em, and it should be rendered as 200x200px on regular desktops. But in IE9-11 it would be rendered as 198.4×198.4px… The effect got worse for iconic images. And sprites. Oh noes! Even SVG images that are made to look best/sharpest at a certain size. So what gives?

It's rounding errors. Again.

For IE, 62.5% is not a valid font-size. It only accepts whole percentages for a font-size. Luckily this limitation goes for font-size only. Instead, IE is trying to be a good boy to round it to 62%. And 16px*62% happens to be 9.92px. Once I realized this, I understood where that odd image size came from. And why text rendered ever-so-slightly "odd" on those IE's.

The solution? I've hinted to it in my previous comment:

On the HTML, font-size:125% -> 20px
On the BODY, font-size:50% -> 10px (exactly).

Or you can use 60% for 12px, 70% for 14px, and so on. As long as it's a whole percentage.

Not totally 100% ontopic with your post (sorry about that), but very much related I'm sure.

(btw, Vasilis, your comment thingy throws away html tags… my previous comment now looks odd. That's why I've capitalized them here like it's 1999 again :P)

VASILIS REPLIED ON 10/04/2014

Again, Martijn, the issues you have only exist because you insist on thinking in pixels. When you start thinking in one em as a base unit, all those complex problems and even more complex solutions are gone.