



## CSS Font-Size: em vs. px vs. pt vs. percent

One of the most confusing aspects of CSS styling is the application of the **font-size** attribute for text scaling. In CSS, you're given four different units by which you can measure the size of text as it's displayed in the web browser. Which of these four units is best suited for the web? It's a question that's spawned a diverse variety of debate and criticism. Finding a definitive answer can be difficult, most likely because the question, itself, is so difficult to answer.

## Meet the Units

1. **“Ems” (em):** The “em” is a scalable unit that is used in web document media. An em is equal to the current font-size, for instance, if the font-size of the document is 12pt, 1em is equal to 12pt. Ems are scalable in nature, so 2em would equal 24pt, .5em would equal 6pt, etc. Ems are becoming increasingly popular in web documents due to scalability and their mobile-device-friendly nature.
2. **Pixels (px):** Pixels are fixed-size units that are used in screen media (i.e. to be read on the computer screen). One pixel is equal to one dot on the computer screen (the smallest division of your screen's resolution). Many web designers use pixel units in web documents in order to produce a pixel-perfect representation of their site as it is rendered in the browser. One problem with the pixel unit is that it does not scale upward for visually-impaired readers or downward to fit mobile devices.
3. **Points (pt):** Points are traditionally used in print media (anything that is to be printed on paper, etc.). One point is equal to 1/72 of an inch. Points are much like pixels, in that they are fixed-size units and cannot scale in size.
4. **Percent (%):** The percent unit is much like the “em” unit, save for a few fundamental differences. First and foremost, the current font-size is equal to 100% (i.e. 12pt = 100%). While using the percent unit, your text remains fully scalable for mobile devices and for accessibility.

## So, What's the Difference?

It's easy to understand the difference between font-size units when you see them in action.

Generally, **1em = 12pt = 16px = 100%**. When using these font-sizes, let's see what happens when you

increase the base font size (using the body CSS selector) from 100% to 120%.

	<code>body { font-size: 100%; }</code>	<code>body { font-size: 120%; }</code>
<code>font-size: 1em</code>	The quick brown fox	The quick brown
<code>font-size: 12pt</code>	The quick brown fox	The quick brown fox
<code>font-size: 16px</code>	The quick brown fox	The quick brown fox
<code>font-size: 100%</code>	The quick brown fox	The quick brown

© KyleSchaeffer.com

As you can see, both the em and percent units get larger as the base font-size increases, but pixels and points do not. It can be easy to set an absolute size for your text, but it's much easier on your visitors to use scalable text that can display on any device or any machine. For this reason, the em and percent units are preferred for web document text.

## Em vs. Percent

We've decided that point and pixel units are not necessarily best suited for web documents, which leaves us with the em and percent units. In theory, both the em and the percent units are identical, but in application, they actually have a few minor differences that are important to consider.

In the example above, we used the percent unit as our base font-size (on the body tag). **If you change your base font-size from percent to ems** (i.e. `body { font-size: 1em; }`), you probably won't notice a difference. Let's see what happens when "1em" is our body font-size, and when the client alters the "Text Size" setting of their browser (this is available in some browsers, such as Internet Explorer).

<code>body = 1em</code>	Text-Size: "Smallest"	Text-Size: "Largest"
<code>font-size: 1em</code>	The quick brown fox jumps over the lazy dog.	The quick
<code>font-size: 100%</code>	The quick brown fox jumps over the lazy dog.	The quick br

© KyleSchaeffer.com

When the client's browser text size is set to "medium," there is no difference between ems and percent. When the setting is altered, however, the difference is quite large. On the "Smallest" setting, ems are much smaller than percent, and when on the "Largest" setting, it's quite the opposite, with ems displaying much larger than percent. While some could argue that the em units are scaling as they are truly intended, in practical application, the em text scales too abruptly, with the smallest text becoming hardly legible on some client machines.

## The Verdict

In theory, the em unit is the new and upcoming standard for font sizes on the web, but in practice, the percent unit seems to provide a more consistent and accessible display for users. When client settings have changed, percent text scales at a reasonable rate, allowing designers to preserve readability, accessibility, and visual design.

**The winner:** percent (%).

## Addendum (January 2011)

It's been a couple years since I wrote this post, and I'd like to sum up the discussion and debate that has happened in that time. Generally, when I create a new design, I will use percent on the `body` element (`body { font-size: 62.5%; }`), and then use the `em` unit to size it from there. As long as the `body` is set using the percent unit, you may choose to use either percent or ems on any other CSS rules and selectors and still retain the benefits of using percent as your base font size. Over the past couple of years, this has really become the standard in design.

Pixels are now considered *acceptable* font size units (users can use the browser's "zoom" feature to read smaller text), although they are starting to cause some issues as a result of mobile devices with very high density screens (some Android and iPhone devices have upwards of 200 to 300 pixels per inch, making your 11- and 12-pixel fonts very difficult to see!). As a result, I will continue to use percent as my base font size in web documents. As always, discussion and debate is encouraged and welcome; thanks for all the great comments over the course of the past two years!

Posted on September 30th, 2008 by Kyle in [Development](#), [User Experience](#).

← [Karate Corners: XHTML/CSS Rounded Corners](#)