## Font-Size Measurements

Determining a unit of measurement to size our text can be a topic of heated debate, even in this day and age.

There are two types of measurement units: **relative lengths** and **absolute lengths**.

* **Relative lengths**: Relative length units specify a length relative to another length. The relative units are: **%,** **em**, **rem**, **vw**, **vh**, **vmin**, **vmax**. They are mainly useful when the output media has dynamic size like mobile device, laptop…
* **Absolute lengths**: are fixed in relation to each other and anchored to some physical measurement. The absolute units are: **px**, **mm**, **cm**, **in**, **pt**, **pc**. They are mainly useful when the output environment is known.
* **mm**: millimeters, e.g. 10mm.
* **cm**: centimeters, e.g. 1cm (= 10mm).
* **in**: inches, e.g. 0.39in ( = 10mm).
* **pt**: point, where 1pt is generally assumed to be 1/72 inch e.g. 12pt.
* **pc**: pica, where 1pc is 12pt
* **px**: pixel, e.g. 14px.

In general, there are issues with all these measurement units. Millimeters, centimeters and inches are inaccurate for a screen-based medium. Points and picas are unreliable since systems can use different dpi settings.

## Meet the Units

Sizing with px

In the early days of the web, we used pixels to size our text. It's reliable and consistent. 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.

Unfortunately, users of Internet Explorer—even in IE9—do not have the ability to change the size of the text using the browser function of increasing or decreasing font size. For those concerned about the usability of their site, this may be a big deal. Recent versions of IE include zooming, which increases the size of everything on the page—a feature that is also available in most other mainstream browsers, too. This has helped mitigate the issue to a degree.

Sizing with em

That whole inability to resize text in IE was a frustration. To get around that, the **em** unit was introduced.

The size of an **em** value is dynamic. When defining the font-size property, an **em** is equal to the size of the font that applies to the parent of the element in question. If you haven't set the font size anywhere on the page, then it is the browser default, which is probably 16px.

So, by default 1em = 16px, and 2em = 32px. If you set a font-size of 20px on the body element, then 1em = 20px and 2em = 40px. In order to calculate the **em** equivalent for any pixel value required, you can use this formula:

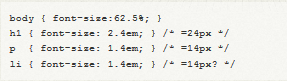


For example, suppose the font-size of the body of the page is set to 1em, with the browser standard of 1em = 16px; if the font-size you want is 12px, then you should specify 0.75em (because 12/16 = 0.75). Similarly, if you want a font size of 10px, then specify 0.625em (10/16 = 0.625); for 22px, specify 1.375em (22/16).

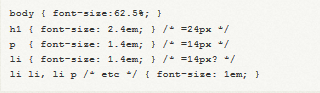
A popular technique to use throughout the document is to set the font-size of the body to 62.5% (that is 62.5% of the default of 16px), which equates to 10px, or 0.625em. Now you can set the font-size for any elements using em units, with an easy-to-remember conversion, by dividing the px value by 10. This way 6px = 0.6em, 8px = 0.8em, 12px = 1.2em, 14px = 1.4em, 16px = 1.6em. For example:



The **em** is a very useful unit in CSS, since it can adapt automatically to the font that the reader chooses to use.



The problem with em-based font sizing is that the font size **compounds**. A list within a list isn't 14px, it's 20px. Go another level deeper and it's 27px! These issues can be worked around by declaring any child elements to use 1em, avoiding the compounding effect.

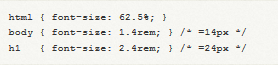


The compounding nature of em-based font-sizing can be frustrating so what else can we do?

Sizing with rem

CSS3 introduces a few new units, including the **rem** unit, which stands for "**root em**".

The **em** unit is relative to the font-size of the parent, which causes the compounding issue. The **rem** unit is relative to the **root**—or the html—element. That means that we can define a single font size on the html element and define all rem units to be a percentage of that.



You can define a base font-size of 62.5% to have the convenience of sizing rems in a way that is similar to using px.

You might be surprised to find that browser support is surprisingly decent: Safari 5, Chrome, Firefox 3.6+, and even Internet Explorer 9 have support for this. The nice part is that IE9 supports resizing text when defined using rems.

What do we do for browsers that don't support rem units? We can specify the fall-back using px, if you don't mind users of older versions of Internet Explorer still being unable to resize the text (well, there's still page zoom in IE7 and IE8). To do so, we specify the font-size using px units first and then define it again using rem units.



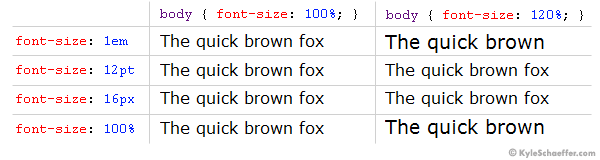
And voila, we now have consistent and predictable sizing in all browsers, and resizable text in the current versions of all major browsers.

Sizing with Percentages

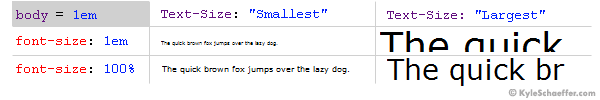
Percentages are fairly obvious in how they work. That is, they work just how you think they will. If a parent has the font-size of 20px and the child has a font-size of 50%, it will come out to 10px. Just like em's the very nature of percentage sizing is that it is relative. It also cascades in the same way, so the very problem described above of the list item applies here as well.

## Em vs. Percent

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).

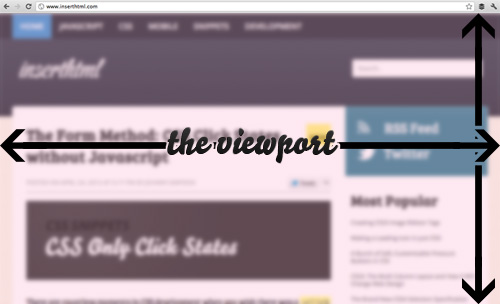


When the client’s browser text size is set to “medium,” there is no difference between ems and percent. When the setting is altered, however, the difference is quite large. On the “Smallest” setting, ems are much smaller than percent, and when on the “Largest” setting, it’s quite the opposite, with ems 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.

vw, vh and vmin

These new CSS3 properties allow you to scale font sizes according to the viewport dimensions.

The viewport is the width and height of the viewable portion of the screen, inside the browser.



There are 3 new viewport related units, which will be incredibly useful for many things. These are **vh, vw** and **vmin**.

The **vw** unit stands for **viewport width** and each unit of **vw** is equal to 1% of the total viewport width. For example:

|  |  |
| --- | --- |
|  | div {       width: 1vw;} |

This would make the div 1% of the total viewport width. It would change as the screen size changed. Similarly, **vh** stands for viewport height, and each unit is equal to 1% of the viewport height. The final unit, **vmin** calculates which is smaller, the horizontal viewport width or the vertical viewport height, and then bases its calculations off that.

For example, if the height of the viewport was smaller than the width, then 1vmin would be 1% of the viewport height.

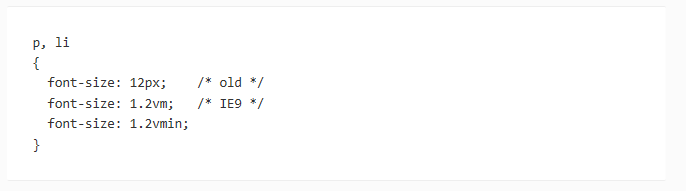
This unit is useful for avoiding compounding in width calculations, just as rem avoids font-size multiplier compounding.

The new units will revolutionize responsive design — text on mobile devices often appears a little large because you’re holding the device closer than a monitor.

Browser support is a little patchy but it’s coming…

* IE10 — full support
* IE9 — supported, but vmin is named “vm”
* Chrome 22+ — full support
* Safari 6 and iOS Safari 6 — full support
* Firefox —version 19
* Blackberry Browser 10 — full support

No word from Opera yet, but I suspect they’re on the case. Again, it may be advisable to use fallbacks for a few years, e.g.



Summary

**Generally, 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 but keep an eye out for **rems** as they will become more widely used.

What follows is a list of suggestions, not absolute rules:

Pixels (px)

**Use for:** hairline borders and general elements when creating fixed-width designs; values for [CSS shadow](http://demosthenes.info/blog/css/shadows) displacement. Avoid using in [@media](http://demosthenes.info/blog/154/CSS-media-queries) breakpoints, as doing so [breaks pages when they are zoomed](http://blog.cloudfour.com/the-ems-have-it-proportional-media-queries-ftw/): use rem or em instead.

**Don’t use for:** typography. (Exception: setting a base font-size in a [CSS reset](http://demosthenes.info/blog/210/CSS-Resets)).

Percentage (%)

**Use for:** making [responsive images](http://demosthenes.info/blog/586/CSS-Fluid-Image-Techniques-for-Responsive-Site-Design) and [containers](http://demosthenes.info/blog/627/Make-A-Responsive-CSS3-Image-Slider); setting height on the body in some cases. Body base font-size.

em, ex

**Use for:** typography, and elements related to typography (margins, for example), with the understanding that em has subtle “gotchas” when used in complex layouts. Consider using [rem](http://demosthenes.info/blog/673/REM-Not-the-Band) as an alternative.

Points and picas

**Use for:** [print stylesheets](http://demosthenes.info/blog/css/print).

**Don’t use for:** anything else.

rem

**Used as:** a more robust and predictable alternative to em and ex, and employed for the same purposes, including @media query breakpoints.

**Don’t use:** if you wish to support IE8 and earlier. Or, use the unit but include a fallback (by providing an alternate measurement in a more common unit before the rem measurement)

Viewport units (vh & vw)

**Use for:** [responsive typography](http://demosthenes.info/blog/739/Creating-Responsive-Hero-Text-With-vw-Units); [“perfect”](http://demosthenes.info/blog/738/Create-Perfect-Responsive-Shapes-With-CSS-vw-Units) responsive containers.

What **not to use** the units for is difficult to determine, as vh & vw are very new and have yet to be fully exploited in web design. Do be aware of their lack of support in IE 8, and allow for fallbacks.

Inches (in) and Centimeters (cm)

**Use with:** print stylesheets, especially page margins

**Don’t use for:** anything else.

Absolute Font Sizing Keywords

Several absolute font-sizing keywords are available. The font size is determined from a browser preset and the element will not inherit its parent’s size.

* font-size: xx-small;
* font-size: x-small;
* font-size: small;
* font-size: medium;
* font-size: large;
* font-size: x-large;
* font-size: xx-large;

Relative Font Sizing Keywords

Two relative font-sizing keywords are available. The font is sized according to its parent element:

* font-size: smaller;
* font-size: larger;

For example, if the parent has a font size of ‘medium’, a value of ‘larger’ will set the element to ‘large’. Other font units are normally altered by a factor of around 1.2 but, again, there is no standard and browser results will differ.

CSS Typography Conversion Reference

Assumptions: The generic browser font size default setting is 16px and the <body> selector font size is set to 1em or 100%.

It’s an approximation, which will depend on font, browser and OS, but it’s a good starting point.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Conversion Table** | | | | | |
| **Points  (for print css)** | **Pixels** | **Ems** | **Percent** | **HTML Size** | **Keyword** |
| 6 pt | 8 px | 0.5 em | 50% |  |  |
| 7 pt | 9 px | 0.55 em | 55% |  | xx-smal |
| 7.5 pt | 10 px | 0.625 em | 62.5% | 1 | x-small |
| 8 pt | 11 px | 0.7 em | 70% | **H6** |  |
| 9 pt | 12 px | 0.75 em | 75% | 2 |  |
| 10 pt | 13 px | 0.8 em | 80% | **H5** | small |
| 10.5 pt | 14 px | 0.875em | 87.5% |  |  |
| 11 pt | 15 px | 0.95 em | 95% |  |  |
| **12 pt** | **16 px** | **1 em** | **100%** | **3**  **H4** | medium |
| 13 pt | 17 px | 1.05 em | 105% |  |  |
| 13.5 pt | 18 px | 1.125 em | 112.5% | 4 |  |
| 14 pt | 19 px | 1.2 em | 120% | **H3** | large |
| 14.5 pt | 20 px | 1.25 em | 125% |  |  |
| 15 pt | 21 px | 1.3 em | 130% |  |  |
| 16 pt | 22 px | 1.4 em | 140% |  |  |
| 17 pt | 23 px | 1.45 em | 145% |  |  |
| 18 pt | 24 px | 1.5 em | 150% | **H2** | x-large |
| 20 pt | 26 px | 1.6 em | 160% | 6 |  |
| 22 pt | 29 px | 1.8 em | 180% |  |  |
| 24 pt | 32 px | 2 em | 200% | **H1**  6 |  |
| 26 pt | 35 px | 2.2 em | 220% |  |  |
| 27 pt | 36 px | 2.25 em | 225% |  |  |
| 28 pt | 37 px | 2.3 em | 230% |  |  |
| 29 pt | 38 px | 2.35 em | 235% |  |  |
| 30 pt | 40 px | 2.45 em | 245% |  |  |
| 32 pt | 42 px | 2.55 em | 255% |  | xx-large |

<http://pxtoem.com/> [PX to EM conversion made simple.](http://pxtoem.com/)

<http://www.smashingmagazine.com/2013/05/17/typographic-design-patterns-practices-case-study-2013/>

Units of measurement for everything else

**Em**

Relative unit that responds to text-size preferences.

Using margins as an example, the em unit scales the margins according to the size of the font of the container/ element or if the container does not have a font-size associated with it uses the nearest font-size.

Let's take this example:

h1 {margin-top:2em}

This sets the top margin for h1 heading to 2 times the height of the h1 font-size.

div.box {margin: 2em; width: 20em;}

Create a box about 20 times the default font size, with a 2x the default font size margin around it. The default font-size is either set in the body or the browser. The body is checked first.

.Heading  
{  
  font-size: 20px;  
  margin-bottom : .05em;  
}

A class named Heading, which sets the font size to 20 Pixels, and the margin-bottom property (which defines the style of the bottom margin of the element that will use this class) to .05em. The nearest font size property is in the same rule, and has a value of 20px, so .05 means one pixel only.

**Percentages**

Relative to the size of the container.

Percent works similarly to em, except that it's more selector-specific.

In the text-indent, margin, padding, and width properties, percentage values are relative to the width of the parent element. You'll remember that an element is contained within another element (for example, a <paragraph> will be inside the <body>). Usually percentage values specify a percentage of the parent.

For example, specifying a width property with a percentage value means the width of an element will be the given percentage of the element which contains it. For instance, p {width: 75%} means that paragraphs will be 75% the width of their container. Often this will mean the body element.

**Review**:

http://www.smashingmagazine.com/2013/05/17/typographic-design-patterns-practices-case-study-2013/