Web Style Sheets
CSS tips & tricks

See also the index of all tips.

On this page:

EM, PX, PT, CM, IN…

CSS offers a number of different units for expressing length. Some have their history in typography, such as point (**pt**) and pica (**pc**), others are known from everyday use, such as centimeter (**cm**) and inch (**in**). And there is also a "magic" unit that was invented specifically for CSS: the **px**. Does that mean different properties need different units?

No, the units have nothing to do with the properties, but everything with the output media: screen or paper.

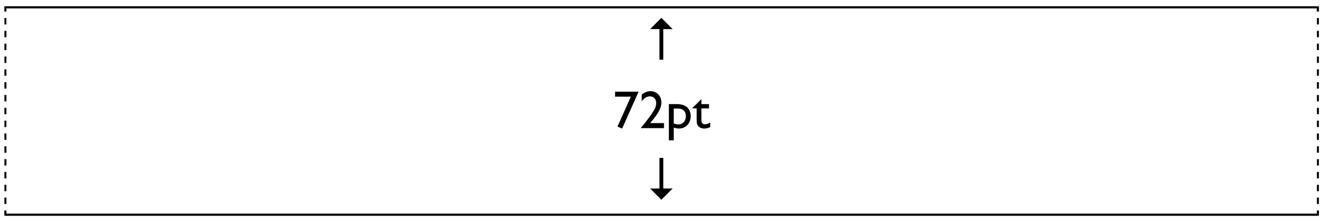
There is no restriction on which units can be used where. If a property accepts a value in **px** (**margin: 5px**) it also accepts a value in inches or centimeters (**margin: 1.2in; margin: 0.5cm**) and vice-versa.

But in general you would use a different set of units for display on screen than for printing on paper. The following table gives the recommended use:

	Recommended	**Occasional use**	**Not recommended**
Screen	em, px, %	ex	pt, cm, mm, in, pc
Print	em, cm, mm, in, pt, pc, %	px, ex	

The relation between the absolute units is as follows: 1in = 2.54cm = 25.4mm = 72pt = 6pc

If you have a ruler handy you can check how precise your device is: here is a box of 1in (2.54cm) high:



The so-called *absolute* units (**cm**, **mm**, **in**, **pt** and **pc**) mean the same in CSS as everywhere else, *but only if your output device has a high enough resolution.* On a laser printer, 1cm should be exactly 1 centimeter. But on low-resolution devices, such as computer screens, CSS doesn't require that. And indeed, the result tends to be different from one device to another and from one CSS implementation to another. It's better to reserve these units for high-resolution devices and in particular for printed output. On computer screens and handheld devices, you'll probably not get what you expect.

In the past, CSS required that implementations display absolute units correctly even on computer screens. But as the number of incorrect implementations outnumbered correct ones and the situation didn't seem to improve, CSS abandoned that requirement in 2011. Currently, absolute units must work correctly only on printed output and on high-resolution devices.

CSS doesn't define what “high resolution” means. But as low-end printers nowadays start at 300 dpi and high-end screens are at 200 dpi, the cut-off is probably somewhere in between.

There is another reason to avoid absolute units for other uses than print: You look at different screens from different distances. 1cm on a desktop screen looks small. But the same on a mobile phone directly in front of your eyes looks big. It's better to use relative units, such as **em**, instead.

The **em** and **ex** units depend on the font and may be different for each element in the document. The **em** is simply the font size. In an element with a 2in font, 1em thus means 2in. Expressing sizes, such as margins and paddings, in **em** means they are related to the font size, and if the user has a big font (e.g., on a big screen) or a small font (e.g., on a handheld device), the sizes will be in proportion. Declarations such as **text-indent: 1.5em** and **margin: 1em** are extremely common in CSS.

The **ex** unit is rarely used. Its purpose is to express sizes that must be related to the x-height of a font. The x-height is, roughly, the height of lowercase letters such as *a, c, m,* or *o.* Fonts that have the same size (and thus the same **em**) may vary wildly in the size of their lowercase letters, and when it is important that some image, e.g., matches the x-height, the **ex** unit is available.

The **px** unit is the magic unit of CSS. It is not related to the current font and usually not relatated to physical centimeters or inches either. The **px** unit is defined to be small but visible, and such that a horizontal 1px wide line can be displayed with sharp edges (no anti-aliasing). What is sharp, small and visible depends on the device and the way it is used: do you hold it close to your eyes, like a mobile phone, at arms length, like a computer monitor, or somewhere in between, like an e-book reader? The **px** is thus not defined as a constant length, but as something that depends on the type of device and its typical use.

To get an idea of the appearance of a **px**, imagine a CRT computer monitor from the 1990s: the smallest dot it can display measures about 1/100th of an inch (0.25mm) or a little more. The **px** unit got its name from those screen pixels.

Nowadays there are devices that could in principle display smaller sharp dots (although you might need a magnifier to see them). But documents from the last century that used **px** in CSS still look the same, no matter what the device. Printers, especially, can display sharp lines with much smaller details than 1px, but even on printers, a 1px line looks very much the same as it would look on a computer monitor. Devices change, but the **px** always has the same visual appearance.

In fact, CSS requires that **1px** must be exactly 1/96th of an inch in all printed output. CSS considers that printers, unlike screens, do not need to have different sizes for **px** in order to print sharp lines. In print media, a px thus not only has the same visual appearance from one device to another, but indeed it is measurably the same (just as the **cm**, **pt**, **mm**, **in** and **pc**, as explained above).

CSS also defines that raster images (such as photos) are, by default, displayed with one image pixel mapping to 1px. A photo with a 600 by 400 resolution will be 600px wide and 400px high. The pixels in the photo thus do not map to pixels of the display device (which may be very small), but map to **px** units. That makes it possible to exactly align images to other elements of a document, as long as you use **px** units in your style sheet, and not **pt**, **cm**, etc.

USE **EM** OR **PX** FOR FONT SIZES

CSS inherited the units **pt** (point) and **pc** (pica) from typography. Printers have traditionally used those and similar units in preference to **cm** or **in**. In CSS there is no reason to use **pt**, use whichever unit you prefer. But there *is* a good reason to use *neither* **pt** *nor any other absolute unit* and only use **em** and **px**.

Here are a few lines of different thickness. Some or all of them may look sharp, but at least the 1px and 2px lines should be sharp and visible:

|0.5pt, |1px, |1pt, |1.5px, |2px

If the first four lines all look the same (or if the 0.5pt line is missing), you are probably looking at a computer monitor that cannot display dots smaller than 1px. If the lines appear to increase in thickness, you are probably looking at this page on a high-quality computer screen or on paper. And if 1pt looks thicker than 1.5px, you probably have a handheld screen.

The magic unit of CSS, the **px**, is a often a good unit to use, especially if the style requires alignment of text to images, or simply because anything that is 1px wide or a multiple of 1px is guaranteed to look sharp.

But for font sizes it is even better to use **em**. The idea is (1) to not set the font size of the BODY element (in HTML), but use the default size of the device, because that is a size that the reader can comfortably read; and (2) express font sizes of other elements in **em**: **H1 {font-size: 2.5em}** to make the H1 2½ times as big as the normal, body font.

The only place where you could use **pt** (or **cm** or **in**) for setting a font size is in style sheets for print, if you need to be sure the printed font is exactly a certain size. But even there using the default font size is usually better.

✎ The **px** unit thus shields you from having to know the resolution of the device. Whether the output is 96 dpi, 100 dpi, 220 dpi or 1800 dpi, a length expressed as a whole number of **px** always looks good and very similar across all devices. But what if you *do* want to know the resolution of the device, e.g., to know if it is safe to use a **0.5px** line? The answer is to check the resolution via Media Queries. Explaining Media Queries is out of scope for this article, but here is a small example:

```
div.mybox { border: 2px solid }
@media (min-resolution: 2dppx) {
  /* Media with 2 or more dots per px */
  div.mybox { border: 1.5px solid }
}
```

MORE UNITS IN CSS

To make it even easier to write style rules that depend only on the default font size, CSS has since 2013 a new unit: the **rem**. The **rem** (for "root em") is the font size of the root element of the document. Unlike the **em**, which may be different for each element, the **rem** is constant throughout the document. E.g., to give P and H1 elements the same left margin, compare this pre-2013 style sheet:

```
p { margin-left: 1em }
h1 { font-size: 3em; margin-left: 0.333em }
```

with the new version:

```
p { margin-left: 1rem }
h1 { font-size: 3em; margin-left: 1rem }
```

Other new units make it possible to specify sizes relative to the reader's window. These are the **vw** and **vh**. The **vw** is 1/100th of the window's width and the **vh** is 1/100th of the window's height. There is also **vmin**, which stands for whichever is the smallest of **vw** and **vh**. And **vmax**. (You can guess what it does.)

Because they are so new, they don't work everywhere yet. But, as of early 2015, several browsers support them.

Bert Bos, style activity lead

Created 12 Jan 2010;
Last updated Sat 01 Oct 2016 05:20:34 AM UTC

LANGUAGES

- Deutsch
- English
- Français
- Polski
- Português
- Русский
- Українська

About the translations