In this section, we are going to address the topic of specifying **units** in CSS for better maintainability. After introducing the different types on units in CSS, we're going to talk about some maintainability concerns. At the end of this section, we're gonna introduce some CSS shorthands and we'll also learn what unitless values and zero values are.

Units can be *absolute* or *relative*. The difference between absolute and relative units is that relative units scale. Therefore, they grow better with breakpoints whenever dealing with responsive design.

Relative units also work better when zooming in or out in the browser. On the other hand, absolute units are units that we are very much used to in everyday life. You think in terms of centimeters, millimeters, inches, or points. Points are inches divided by 72. Another type of absolute units that's sought for developers frequently used is pixels.

Note though that pixels behave as absolute units, however, they depend on the dot per inch ratio of your device. For maintainability reasons, most of the times it was recommended to use relative units. One popular relative unit is `em`. `em`s are defined with respect to the font size of the element.

For example if you're using 20-pixel phones and the metric is 1.5, in that case the distance itself that we defined is 30 pixels. Changing the font size also changes the size of the container. There is one major problem with units: it is very hard to calculate them sometimes. The need arises that whenever we write the same styling options inside the CSS, we should see the exact same style inside the dom element as well. `em`s do not provide this feature, and for this reason we are going to use relative EMs (`rem`).

Relative units are relative to the font size of the root element in your document, which is the `html` element. If it's ten pixels then `2.5rem` will equal to `25px`. Even though REMs might be the universal solution for specifying your distances, sometimes you have to specify other units as well.

For instance, when you have a container and you have a couple of columns, it makes perfect sense to use percentages. Percentages are units that define the size of their element with respect to the size of the parent element. `50%` means that the size of the child element is one-half of the size of the parent element.

- We use `em` whenever we want to compare our units to the element's font size. Sometimes we might want to do that, especially when we change the font size dynamically.
- We use `rem` whenever we want to specify scalable sizes in general, that are easy to calculate, easy to maintain. One benefit of REMs is that they don't depend on the font size changes in the container.
- Whenever you have multiple columns it makes perfect sense to use percentages. Percentages always fit regardless of the size of the viewport.

We have some other more exotic relative units, for instance `vh` and `vw` that are similar to percentages, except that they are not relative to the container element, but relative to the size of the viewport.

`vmin` and `vmax` specify the minimum and the maximum of the height and width relative units of the viewport, which can be used, for example, if you want to fit a square inside a mobile device, for example, or if you want to zoom into that square so that it occupies the whole screen.

There are also some typographical relative units, for instance `ch` shows the width of the `0` character and `ex` means the height of the lower case `x` character which is normally one half. These are using typography, if you haven't heard of them, you're most likely not going to use these units.