ADVANCED UI

Measurements in css

Introduction

Measuring things is as important an aspect of web design, as any other. The fact that we have at least 10 different measurement units in CSS speaks for itself.
Each unit serves its own specific purpose and makes web pages more flexible for a variety of devices. Once you become acquainted with all these units, you will be able to size your elements more precisely. In this tutorial, we will take a look at the different units that are available in CSS and I'll discuss which units to use in which situations, and how to use them.

Absolute length units

Absolute units are a digital representation of actual measurements in the physical world. These units are not related to the size of the screen or its resolution. As a result, absolute length units are not as well suited for use on digital devices or when the resolution is not known. These units are more appropriate when you are designing for physical media such as print.
Absolute units include:

- cm (centimeters)
- mm (millimeters)
- in (inches)
- pc (picas)
- pt (points)
- px (pixels)

Note that the editor’s draft of the spec also includes the quarter-millimeter (q) unit, but this is new and doesn’t seem to have any browser support.

You might observe that while using absolute lengths there are differences between the same values of a particular unit on different screens. This is because of the difference in the DPI (dots per inch) for a screen. Higher resolution screens have a higher DPI compared to smaller resolution screens, thus making the image or text look smaller.

The most widely used of all absolute units is the pixel (px). A pixel is generally understood to be a single dot on the screen, although it is technically more complex than that. It is the smallest unit of measurement and usually the unit used as a benchmark for the other units.

Other units like inch, millimeter, and centimeter represent the physical size of these units. The point (pt) unit represents 1/72 of an inch and the pica (pc) represents 1/6 of an inch.

Relative length units

Relative units, as the name suggests, don’t have fixed values. Their values are relative to some other predefined value or feature. Relative units make it easy to properly size elements since we can relate their width, height, font-size, etc. to some other base parameter.

These units are usually recommended when you creating responsive layouts and are preferred for digital media. Their value can be relative to the fonts you are using, or to the height and the width of the view window of a screen.

Relative units include:

- ex (x-height)
- ch (character)
- em (named after print ems, but not the same)
- rem (root em)

Let’s take a look at each of these in more detail.

X-height (ex) and Character (ch) units

The ex unit is rarely used in development. `1ex` is equal to the size of the lowercase ‘x’ in the font being used. In most cases, the value of 1ex is nearly equal to 0.5em. However this changes from one font to another. This unit can be considered the vertical equivalent of `em`.

```
p {
  font-size: 2ex;
}
```

The character (ch) unit is related to the ‘0’ character. `1ch` is the advance measure of the character ‘0’ in a font.

```
p {
  margin: 2ch;
}
```

Em units

The em unit has a value equal to the font size of the base element or the parent element. For instance, if the font size of parent element is `20px` then the value of `1em` will compute to `20px` for all immediate child elements. The font size of a child element can be increased or decreased easily based on knowledge of the base unit. The number need not be a whole number.

Using `em` makes it easy for us to keep font sizes of various elements in a fixed ratio. For example, if the value of `font-size` for a parent element is `50px`, setting the font size of a child element to `2em` will be the same as setting it to `100px`. Similarly, setting it to `0.5em` will make the font size of the child element `25px`.

Sometimes, however, `em` units produce undesired results in the case of nested elements. This is because the `em` value takes the value of the immediate parent tag, so each nested child will have the same `em` value, but a different computed value, because the computed value is always relative to its immediate parent. Thus it makes it difficult if we need to set the value of the current element in ratio to a parent element that is not immediate or not the root parent.

Rem units

This is where the rem unit comes in handy. The value of a `rem` always stays equal to the font size of the root element (which is the `html` element in HTML documents). The `rem` demo is similar to that of the`em` units. Here is the CSS:

Viewport-relative length units

Viewport-relative lengths are based on the width and height of the view window or the viewport. A view window or viewport is the visible area or the framed space on the screen.

Viewport-relative units include:

- vh (viewport height)
- vw (viewport width)
- vmin (viewport minimum)
- vmax (viewport maximum)

Viewport Height (vh) and Viewport Width (vw)

The vh unit is related to the height of the viewport. The value of `vh` is equal to 1/100th of the height of the viewport. This unit is useful if we want to scale elements based on the height of the browser window. For example, if the height of the viewport is `400px`, `1vh` equals `4px`. If the viewport height is `800px`,`1vh` equals `8px`.

Just as `vh` is related to the height of the window, vw units are related to the width of the viewport. The value of `1vw` can therefore be deduced similarly. This means that `1vw` is equal to 1/100th of the width of the viewport. For example, if the width of the window is `1200px`, `1vw` will be `12px`. The CSS for setting the width, height, and padding of an element in viewport units is:

```
div {
  height: 80vh;
  width: 95vw;
  padding: 1vw;
}
```

Viewport minimum (vmin) and Viewport maximum (vmax) units

The vmin unit is related to the side of the viewport with the lower value so it can be either the height or the width. The value of `1vmin` is equal to 1/100th of the side with the smallest length. For example, if the dimensions of the viewport are 500 x 700 then the value of `1vmin` equals `5px`. If the dimensions were 1000 x 700, the value would be 7px.

Conversely, vmax takes into consideration the maximum value of the viewport. The relation factor is again 1/100, so `1vmax` is 1/100th of the edge with the higher value. Taking the same examples as above, if the dimensions of the viewport are 500 x 700 then the value of 1vmax is equal to 7px. In case the dimension was 1000 x 700, the value would be 10px. This is the CSS to set width and height in `vmin`and `vmax`:

```
div {
  height: 80vmin;
  width: 95vmax;
  padding: 1vmax;
}
```

Browser support

em, ex, px, cm, mm, in, pt, and pc

Supported in all browsers, including old IE.

ch

Firefox, Chrome 27+, IE 9+, Safari 7+, and Opera 20+.

rem

All in-use browsers including IE9+. If you need more support, here is a polyfill.

vw, vh, and vmin

Chrome 20+, IE 9+, Firefox 19+, Safari 6+, Opera 20+. One thing worth remembering is that in Internet Explorer `vmin` is supported by the non-standard `vm` syntax. For a polyfill you can try vminpoly or use -prefix-free with a plugin.

vmax

Chrome 20+, Firefox 19+, and Opera 20+, and Safari 6.1+. No support in IE.