Sizing in CSS

This reference material will cover some of the common CSS properties used to size elements.

It's easy to understand this in the context of sizing different typography elements, but keep in mind thats all of these units can be applied to any property that expects a numerical size to be set (`height`, `width`, `margin`, `padding`, `font-size`, `border-width`, etc.).

Each class in the following example uses a different sizing unit to the elements where the class is applied. Each unit behaves a bit differently, and it's important to understand the difference between them.

```
.font-small {
    font-size: 80%;
}

.pixels {
    font-size: 16px;
}

.ems {
    font-size: 1em;
}

.rems {
    font-size: 1rem;
}

.percent {
    font-size: 100%;
}
```

Absolute Sizes

Absolute units are, well, absolute. They do not scale with the viewport size or display, and they are a bit antiquated at this point. They still have their use from time to time and you will still encounter them in web development, so it is important to understand what they are.

Pixels

> Pixels (`px`) are absolute units, meaning that an element that has a size defined in `px` will always be that size, regardless of whether you're viewing it on an 5" phone display or a 65" 4K television.
>
> You almost *always* want to use *relative sizes* because there are so many different ways to consume content now. When everyone was using CRT desktop monitors before the proliferation of smart devices, it was acceptable to use pixels because they were the same size on all of the monitors - and everyone's monitor was pretty much the same. Now, in order to provide a consistent and uniform user experience across as many devices as possible, *you need to use relative units*.
>
> There are other, less common, absolute units in CSS, but we typically only use pixels (since that's what most display resolutions and browser windows are measured in). These would be `mm`, `cm`, `in`, `pt`, `pc`. You may encounter these in other formats (like Photoshop), so keep in mind the *conversion* that would need to be calculated between absolute units and relative units when working with designers.

Certain elements in HTML have default sizes in pixels in the browser. For example, the `body` has a default typography size of 1
This is important to know because *relative units* sometimes use this "root" size to determine their relative scale.

Relative Sizes (best for responsive design)

Relative sizes are just that: *relative*. We've already discussed the importance of using relative sizes; now, let's discuss what they are, and how they're different.

ems and rems

`em` and `rem` are the units you will use to assign a relative size to an element. These are relative to the current element's font-size, which is either inherited by the parent element (`em`) or determined by the root element (`rem`, thus the "r"). The numeric value of the unit will be the multiplier to determine the element's size relative to its reference point (the parent or the root element).

What does that mean? `font-size: 1em;` and `font-size: 1rem` are not always equivalent. `1em` is going to be 1x the scale of the parent element's font-size; `1rem` will be 1x the scale of the *root* font-size (e.g., 16px, if it isn't given a different size). `2rem` would be `32px`.

This can be confusing at first. The best way to understand these units (and their differences) is to actually use them. Replicate what Matt & Will did in the previous video, and then go change it, refresh those changes in your browser (or change them in the Dev Tools Inspector for live changes), and you'll begin to develop an intuitive understanding of these units.

Viewport units

Viewport units are newer, but they've become ubiquitous in web development due to their usefulness. The **viewport** is the viewable area of the browser window.

You can easily set the width and height of elements based on the *viewport* (or what is viewable of the window element) size. For example, setting the size of something based on the viewport height (`vh`) would be a fraction of the viewport height. `height: 1vh;` is equivalent to 1/100th of the total viewport height. The same would be true for viewport width (`vw`). Setting these values to `100vh` or `100vw` would set either the height or width, respectively, to the full width of the viewport's height or width.

Percentages

If you want an element to always be sized relative to the parent container's *width* (THIS IS IMPORTANT and different from using viewport units) then you could use a percentage value.

This is similar to using `em`s, but keep in mind that percentages are always relative to the width of the parent container. If your `container` takes up the full width of the window, and you then set a child of that container to `width: 80%;`, then that child element will always take up 80% of the container's width.

Again, the easiest way to understand this is to play around with it. There are also several examples given in the additional resources link below.

Additional Resources

- MDN: CSS Values and Units (sizing)