Opacity and Alpha

All of the colors we’ve seen so far have been opaque, or non-transparent. When we overlap two opaque elements, nothing from the bottom element shows through the top element. In this exercise, we’ll change the *opacity*, or the amount of transparency, of some colors so that some or all of the bottom elements are visible through a covering element.

To use opacity in the HSL color scheme, use hsla instead of hsl, and four values instead of three. For example:

color: hsla(34, 100%, 50%, 0.1);

The first three values work the same as hsl. The fourth value (which we have not seen before) is the *alpha*. This last value is sometimes called opacity.

Alpha is a decimal number from zero to one. If alpha is zero, the color will be completely transparent. If alpha is one, the color will be opaque. The value for half-transparent would be 0.5.

You can think of the alpha value as, “the amount of the background to mix with the foreground”. When a color’s alpha is below one, any color behind it will be blended in. The blending happens for each pixel; no blurring occurs.

The RGB color scheme has a similar syntax for opacity, rgba. Again, the first three values work the same as rgb and the last value is the alpha. Here’s an example:

color: rgba(234, 45, 98, 0.33);

A little unconventional, but still worth mentioning is how hex colors can also have an alpha value. By adding a two-digit hexadecimal value to the end of the six-digit representation (#52BC8280), or a one-digit hexadecimal value to the end of the three-digit representation (#F003), you can change the opacity of a hexadecimal color. Hex opacity ranges from 00 (transparent) to FF (opaque).

Alpha can only be used with HSL, RGB, and hex colors; we cannot add the alpha value to name colors like green.

There is, however, a named color keyword for zero opacity, transparent. It’s equivalent to rgba(0, 0, 0, 0), and it’s used like any other color keyword:

color: transparent;