Improved Styling with CSS

Darkhan Zhursin
daratechn@gmail.com

Color

CSS supports a wide variety of colors. These include *named colors*, like `blue`, `black`, and `limegreen`, along with colors described by a numeric value. Using a numeric system allows us to take advantage of the whole spectrum of colors that browsers support. In this lesson, we're going to explore all the color options CSS offers.

Colors in CSS can be described in three different ways:

- *Named colors* — English words that describe colors, also called *keyword colors*
- *RGB* — numeric values that describe a mix of red, green, and blue
- *HSL* — numeric values that describe a mix of hue, saturation, and lightness

Foreground vs Background

Before discussing the specifics of color, it’s important to make two distinctions about color. Color can affect the following design aspects:

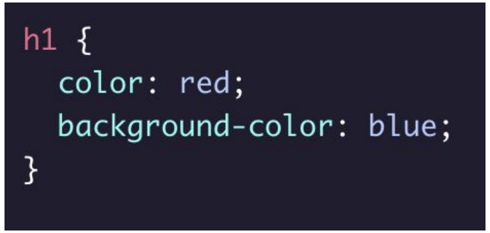
- The foreground color
- The background color

Foreground color is the color that an element appears in. For example, when a heading is styled to appear green, the *foreground color* of the heading has been styled.

Conversely, when a heading is styled so that its background appears yellow, the *background color* of the heading has been styled

In CSS, these two design aspects can be styled with the following two properties:

- `color` - this property styles an element’s foreground color.
- `background-color` - this property styles an element’s background color.

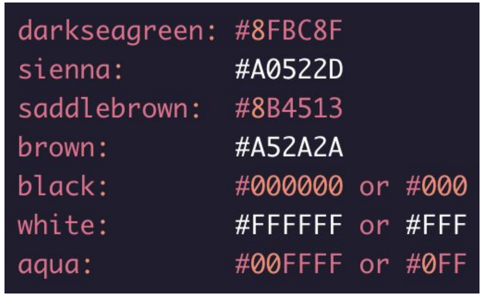


```
h1 {
  color: red;
  background-color: blue;
}
```

In the example above, the text of the heading will appear in red, and the background of the heading will appear blue.

Hexadecimal

One syntax that we can use to specify colors is called *hexadecimal*. Colors specified using this system are called *hex colors*. A hex color begins with a hash character (`#`) which is followed by three or six characters. The characters represent values for red, blue and green.



```
darkseagreen: #8FBC8F
sienna:       #A0522D
saddlebrown:  #8B4513
brown:        #A52A2A
black:        #000000 or #000
white:        #FFFFFF or #FFF
aqua:         #00FFFF or #0FF
```

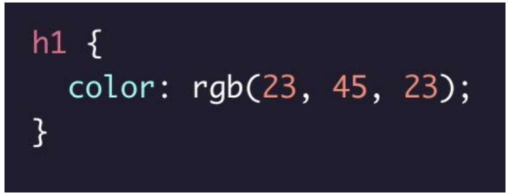
In the example above, you may notice that there are both letters and numbers in the values. This is because the hexadecimal number system has 16 digits (0-15) instead of 10 (0-9) like in the standard decimal system. To represent 10-15, we use A-F. Here is a list of many different colors and their hex values.

Notice that `black`, `white`, and `aqua` are all represented with both three characters and six characters. This can be done with hex colors whose number pairs are the same characters. In the example above, `aqua` can be represented as `#0FF` because both of the first two characters are `0` and the second and third pairs of characters are both `F`s. Keep in mind that all three character hex colors can be represented with six characters (by repeating each character twice) but the same is not true in reverse.

You can include hex colors just as you would include named colors: `background-color: #9932cc;`, and the letters can be uppercase or lowercase.

RGB Colors

There is another syntax for representing RGB values, commonly referred to as “RGB value” or just “RGB”, that uses decimal numbers rather than hexadecimal numbers, and it looks like this:



```css
h1 {
  color: rgb(23, 45, 23);
}
```

Each of the three values represents a color component, and each can have a decimal number value from 0 to 255. The first number represents the amount of red, the second is green, and the third is blue. These colors are exactly the same as hex, but with a different syntax and a different number system.

In general, hex and RGB color representations are equivalent. Which you choose is a matter of personal taste. That said, it’s good to choose one and be consistent throughout your CSS, because it’s easier to compare hex to hex and RGB to RGB.

Hex and RGB

The hexadecimal and rgb color system can represent many more colors than the small set of CSS named colors. We can use this new set of colors to refine our web page’s style.

In both hex and RGB, we have three values, one for each color. Each can be one of 256 values. Specifically, `256 * 256 * 256 = 16,777,216`. That is the amount of colors we can now represent. Compare that to the roughly 140 named CSS colors!

Hue, Saturation, and Lightness

The RGB color scheme is convenient because it’s very close to how computers represent colors internally. There’s another equally powerful system in CSS called the hue-saturation-lightness color scheme, abbreviated as *HSL*.

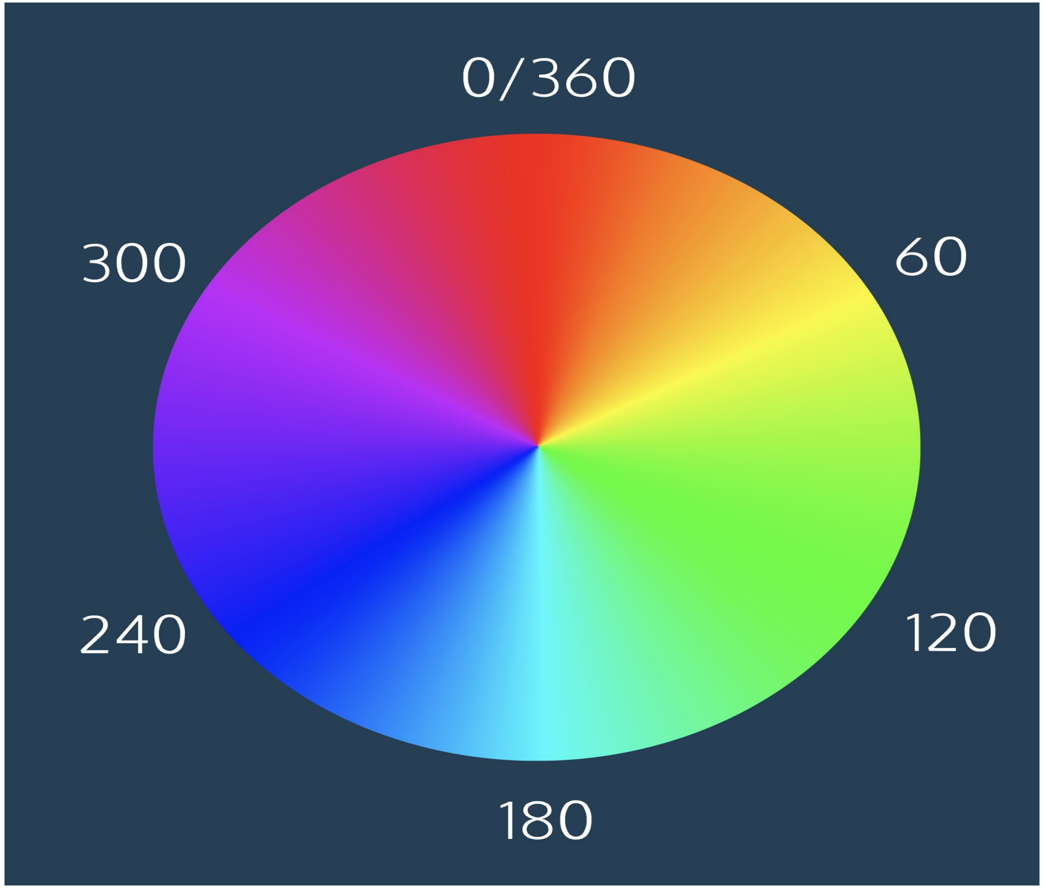
The syntax for HSL is similar to the decimal form of RGB, though it differs in important ways. The first number represents the degree of the hue, and can be between 0 and 360. The second and third numbers are percentages representing saturation and lightness respectively. Here is an example: color: hsl(120%, 60%, 70%);

Hue is the first number. It refers to an angle on a color wheel. Red is 0 degrees, Green is 120 degrees, Blue is 240 degrees, and then back to Red at 360. You can see an example of a color wheel below.

Saturation refers to the intensity or purity of the color. The saturation increases towards 100% as the color becomes richer. The saturation decreases towards 0% as the color becomes grayer.

Lightness refers to how light or dark the color is. Halfway, or 50%, is normal lightness. Imagine a sliding dimmer on a light switch that starts halfway. Sliding the dimmer up towards 100% makes the color lighter, closer to white. Sliding the dimmer down towards 0% makes the color darker, closer to black.

HSL is convenient for adjusting colors. In RGB, making the color a little darker may affect all three color components. In HSL, that’s as easy as changing the lightness value. HSL is also useful for making a set of colors that work well together by selecting various colors that have the same lightness and saturation but different hues.


0/360
60
120
180
240
300

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);

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;

Typography

In this lesson, we’ll focus on *typography*, the art of arranging text on a page. We’ll look at:

- How to style and transform fonts.
- How to lay text out on a page.
- and how to add external fonts to your web pages.

Font Family

h1 { font-family: Arial; }

In the example above, the font family for all `<h1>` heading elements have been set to Arial.

Let's talk about some things to keep in mind when setting `font-family` values.

Multi-Word Values

When specifying a typeface with multiple words, like Times New Roman, it is recommended to use quotation marks (`' '`) to group the words together, like so: h1 { font-family: 'Times New Roman'; }

Web Safe Fonts

There is a selection of fonts that will appear the same across all browsers and operating systems. These fonts are referred to as *web safe fonts*. You can check out a complete list of web safe fonts here.

Fallback Fonts and Font Stacks

Web safe fonts are good *fallback fonts* that can be used if your preferred font is not available.

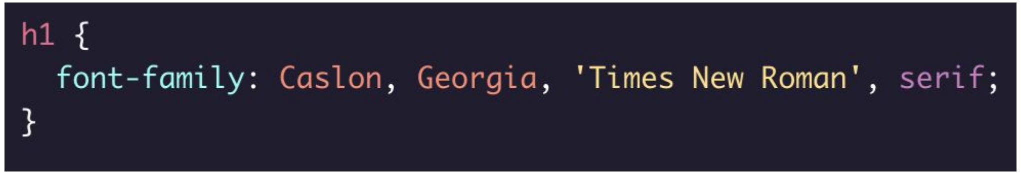
h1 { font-family: Caslon, Georgia, 'Times New Roman'; }

In the example above, Georgia and Times New Roman are fallback fonts to Caslon. When you specify a group of fonts, you have what is known as a *font stack*. A font stack usually contains a list of similar-looking fonts. Here, the browser will first try to use the Caslon font. If that's not available, it will try to use a similar font, Georgia. And if Georgia is not available, it will try to use Times New Roman.

Serif and Sans-Serif

You may be wondering what features make a font similar to another font. The fonts Caslon, Georgia, and Times New Roman are *Serif* fonts. Serif fonts have extra details on the ends of each letter, as opposed to *Sans-Serif* fonts, which do not have the extra details.

`serif` and `sans-serif` are also keyword values that can be added as a final fallback font if nothing else in the font stack is available.



```css
h1 {
  font-family: Caslon, Georgia, 'Times New Roman', serif;
}
```

In this final example, the font stack has 4 fonts. If the first 3 fonts aren’t available, the browser will use whatever serif font is available on the system.

SERIF

Serif fonts have extra details on the ends of the main strokes of the letters. These strokes are called serifs.

im

Serif fonts are traditionally used in print designs, as they have been considered easier to read in long paragraphs of text

SANS SERIF

Sans serif fonts lack those extra strokes on the ends of letters and have flat ends. This gives them a cleaner, more modern look.

im

Since screens have a lower resolution than print, sans serifs can look better and be easier to read.

Font Weight

In CSS, the `font-weight` property controls how bold or thin text appears. It can be specified with keywords or numerical values.

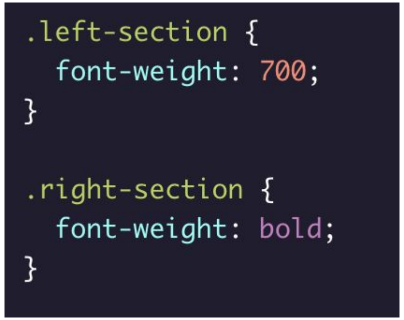
Keyword Values

The `font-weight` property can take any one of these keyword values:

- `bold`: Bold font weight.
- `normal`: Normal font weight. This is the default value.
- `lighter`: One font weight lighter than the element's parent value.
- `bolder`: One font weight bolder than the element's parent value

Font weight: Numerical Values

Numerical values can range from 1 (lightest) to 1000 (boldest), but it is common practice to use increments of 100. A font weight of `400` is equal to the keyword value `normal`, and a font weight of `700` is equal to `bold`.



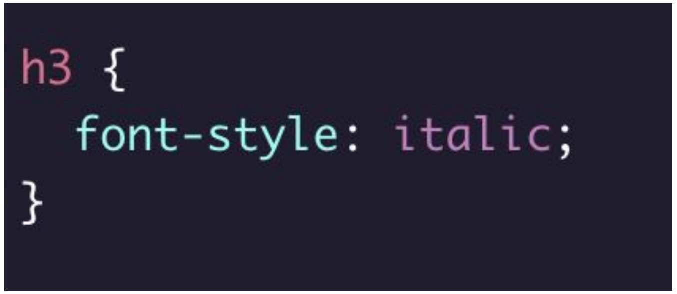
```css
.left-section {
  font-weight: 700;
}

.right-section {
  font-weight: bold;
}
```

In the example above, text in elements of both `.left-section` and `.right-section` classes will appear bold.

It's important to note that not all fonts can be assigned a numeric font weight, and not all numeric font weights are available to all fonts. It's a good practice to look up the font you are using to see which `font-weight` values are available.

Font Style

You can also italicize text with the `font-style` property.

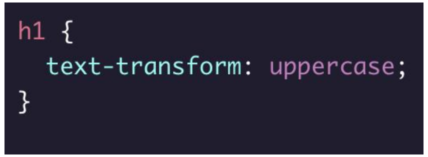


```
h3 {
  font-style: italic;
}
```

The `italic` value causes text to appear in italics. The `font-style` property also has a `normal` value which is the default.

Text Transformation

Text can also be styled to appear in either all uppercase or lowercase with the `text-transform` property.



```
h1 {
  text-transform: uppercase;
}
```

The code in the example above formats all `<h1>` elements to appear in `uppercase`, regardless of the case used for the heading within the HTML code. Alternatively, the `lowercase` value could be used to format text in all lowercase.

Since text can be directly typed in all uppercase or lowercase within an HTML file, what is the point of a CSS rule that allows you to format letter case?

Depending on the type of content a web page displays, it may make sense to always style a specific element in all uppercase or lowercase letters. For example, a website that reports breaking news may decide to format all `<h1>` heading elements such that they always appear in all uppercase, as in the example above. It would also avoid uppercase text in the HTML file, which could make code difficult to read.

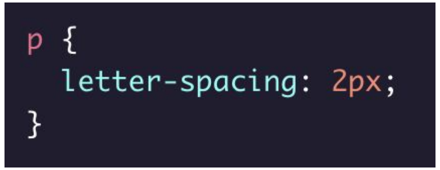
Text Layout

You've learned how text can be defined by font family, weight, style, and transformations. Now you'll learn about some ways text can be displayed or laid out within the element's container.

Letter Spacing

The `letter-spacing` property is used to set the horizontal spacing between the individual characters in an element. It's not common to set the spacing between letters, but it can sometimes help the readability of certain fonts or styles. The `letter-spacing` property takes length values in units, such as `2px` or `0.5em`.

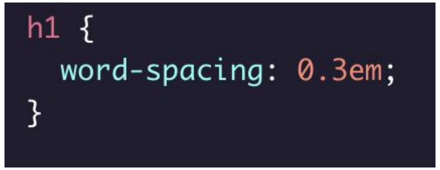
In the example, each character in the paragraph element will be separated by 2 pixels.



```css
p {
  letter-spacing: 2px;
}
```

Word Spacing

You can set the space between words with the `word-spacing` property. It's also not common to increase the spacing between words, but it may help enhance the readability of bolded or enlarged text. The `word-spacing` property also takes length values in units, such as `3px` or `0.2em`.

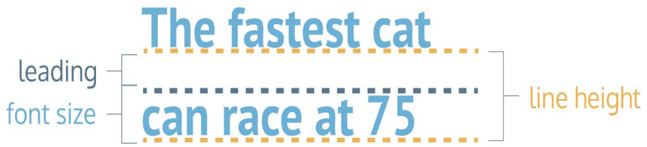


```css
h1 {
  word-spacing: 0.3em;
}
```

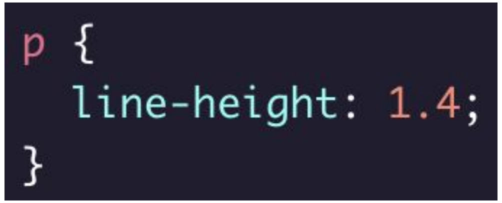
In the example above, the word spacing is set to `0.3em`. For word spacing, using `em` values are recommended because the spacing can be set based on the size of the font.

Line Height

We can use the `line-height` property to set how tall we want each line containing our text to be. Line height values can be a unitless number, such as `1.2`, or a length value, such as `12px`, `5%` or `2em`.



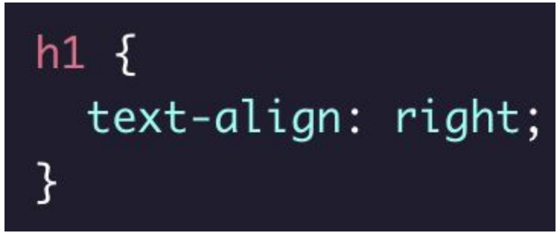
In the example, the height between lines is set to `1.4`. Generally, the unitless value is preferred since it is responsive based on the current font size. In other words, if the `line-height` is specified by a unitless number, changing the font size will automatically readjust the line height.



```
p {
  line-height: 1.4;
}
```

Text Alignment

The `text-align` property aligns text to its parent element.



```
h1 {
  text-align: right;
}
```

Web Fonts

Previously, we learned about web safe fonts, a group of fonts supported across browsers and operating systems. However, the fonts you can use for your website are limitless—*web fonts* allow you to express your unique style through a multitude of different fonts found on the web.

Free font services, like Google Fonts and Adobe Fonts, host fonts that you can link to from your HTML document with a provided `<link>` element.

You can also use fonts from paid font distributors like fonts.com by downloading and hosting them with the rest of your site's files. You can create a `@font-face` ruleset in your CSS stylesheet to link to the relative path of the font file.

Both techniques for including web fonts into your site allow you to go beyond the sometimes "traditional" appearance of web safe fonts. In the next two exercises, you'll learn exactly how to use each of these techniques!

Web Safe Fonts	Web Fonts		
Arial	BANGERS	Chewy	CODYSTAR
Verdana	CREEPSTER	Lobster	Dancing Script
Tahoma	Indie Flower	Pacifico	PERMANENT MARKER
Trebuchet MS	Shadows Into Light	Cutive Mono	Source Code Pro
Times New Roman	Space Mono	VT323	Montserrat Alternates
Georgia	Orbitron	Oswald	Poppins
Courier New	Raleway	Fauna One	Kiwi Maru
Brush Script MT	Merriweather	New Tegomin	SPECTRAL SC