

Learn the Basics

CSS or Cascading Style Sheets, is a presentation language created to style the appearance of content - Using for example, fonts or colors.

The HTML and CSS are independent of one another and should remain that way. CSS should not be written inside of an HTML document and vice versa. As a rule, HTML will always represent content, and CSS will always represent the appearance of that content.

Selectors

As elements are added to a web page, they may be styled using CSS. A selector designates exactly which element or elements within our HTML to target and apply styles to. Selector may include a combination of different qualifiers to select unique elements, all depending on how specific we wish to be. For example, we may want to select every paragraph on a page, or we may want to select only a specific paragraph on a page.

Selectors generally target an attribute value, such as an id or class value, or target the type of element, such as `<h1>` or `<p>`.

Within CSS, selectors are followed with curly brackets, `{}`, which encompass the styles to be applied to the selected element. The selector here is targeting all `<p>` elements

```
p {...}
```

Properties

Once an element is selected, a property determines the styles that will be applied to that element. Property names fall after a selector, within the curly brackets, `{}`, and immediately preceding a colon, `:`. There are numerous properties we can use, such as background, color, font-size, height, and width, and new properties are often added. In the following code, we are defining the color and font-size properties to be applied to all `<p>` elements.

```
p{  
  color: ...;  
  font-size: ...;
```

```
}
```

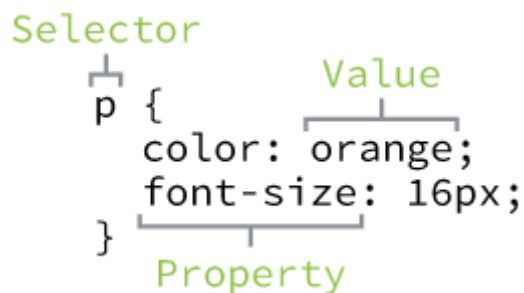
Values

So far we've selected an element with a selector and determined what style we'd like to apply with a property. Now we can determine the behavior of that property with a value. Values can be identified as the text between the colon, :, and semicolon, ;. Here we are selecting all <p> elements and setting the value of the color property to be orange and the value of the font-size property to be 16 pixels.

```
p {  
  color: orange;  
  font-size: 16px;  
}
```

To review, in CSS our rule set begins with the selector, which is immediately followed by curly brackets. Within these curly brackets are declarations consisting of property and value pairs. Each declaration begins with a property, which is followed by a colon, the property value, and finally a semicolon.

It is a common practice to indent property and value pairs within the curly brackets. As with HTML, these indentations help keep our code organized and legible.



A diagram illustrating the components of a CSS rule set. The text `p { color: orange; font-size: 16px; }` is shown with brackets and labels. A bracket above the `p` is labeled "Selector". A bracket above the `orange` in `color: orange;` is labeled "Value". A bracket below the `color: orange;` and `font-size: 16px;` lines is labeled "Property".

Knowing a few common terms and the general syntax of CSS is a great start, but we have a few more items to learn before jumping in too deep. Specifically, we need to take a closer look at how selectors work within CSS.

Working with Selectors

Selectors, as previously mentioned, indicate which HTML elements are being styled. It is important to fully understand how to use selectors and how they can be leveraged. The

first step is to become familiar with the different types of selectors. We'll start with the most common selectors: *type*, *class*, and *ID* selectors.

Type Selectors

Type selectors target elements by their element type. For example, should we wish to target all division elements, `<div>`, we would use a type selector of `div`. The following code shows a type selector for division elements as well as the corresponding HTML it selects.

```
div { ... }
```

Class Selectors

Class selectors allow us to select an element based on the element's class attribute value. Class selectors are a little more specific than type selectors, as they select a particular group of elements rather than all elements of one type.

Class selectors allow us to apply the same styles to different elements at once by using the same class attribute value across multiple elements.

Within CSS, classes are denoted by a leading period, `.`, followed by the class attribute value. Here the class selector will select any element containing the class attribute value of `awesome`, including both division and paragraph elements.

```
.awesome { ... }
```

```
<div class="awesome">...</div>  
<p class="awesome">...</p>
```

ID Selectors

ID selectors are even more precise than class selectors, as they target only one unique element at a time. Just as class selectors use an element's class attribute value as the selector, ID selectors use an element's id attribute value as a selector.

Regardless of which type of element they appear on, id attribute values can only be used once per page. If used they should be reserved for significant elements.

Within CSS, ID selectors are denoted by a leading hash sign, #, followed by the id attribute value. Here the ID selector will only select the element containing the id attribute value of shayhowe.

```
#shayhowe { ... }
```

```
<div id="shayhowe">...</div>
```

Additional Selectors

Selectors are extremely powerful, and the selectors outlined here are the most common selectors we'll come across. These selectors are also only the beginning. Many more [advanced selectors](#) exist and are readily available. When you feel comfortable with these selectors, don't be afraid to look into some of the more advanced selectors.

All right, everything is starting to come together. We add elements to a page inside our HTML, and we can then select those elements and apply styles to them using CSS. Now let's connect the dots between our HTML and CSS, and get these two languages working together.

Referencing CSS

In order to get our CSS talking to our HTML, we need to reference our CSS file within our HTML. The best practice for referencing our CSS is to include all of our styles in a single external style sheet, which is referenced from within the <head> element of our HTML document. Using a single external style sheet allows us to use the same styles across an entire website and quickly make changes sitewide.

Other options for referencing CSS include using internal and inline styles. You may come across these options in the wild, but they are generally frowned upon, as they make updating websites cumbersome and unwieldy.

To create our external CSS style sheet, we'll want to use our text editor of choice again to create a new plain text file with a .css file extension. Our CSS file should be saved within the same folder, or a subfolder, where our HTML file is located.

Within the <head> element of the HTML document, the <link> element is used to define the relationship between the HTML file and the CSS file. Because we are linking to

CSS, we use the `rel` attribute with a value of `stylesheet` to specify their relationship. Furthermore, the `href` (or hyperlink reference) attribute is used to identify the location, or path, of the CSS file.

Consider the following example of an HTML document `<head>` element that references a single external style sheet.

```
<head>
  <link rel="stylesheet" href="main.css">
</head>
```

In order for the CSS to render correctly, the path of the `href` attribute value must directly correlate to where our CSS file is saved. In the preceding example, the `main.css` file is stored within the same location as the HTML file, also known as the root directory.

If our CSS file is within a subdirectory or subfolder, the `href` attribute value needs to correlate to this path accordingly. For example, if our `main.css` file were stored within a subdirectory named `stylesheets`, the `href` attribute value would be `stylesheets/main.css`, using a forward slash to indicate moving into a subdirectory.

At this point our pages are starting to come to life, slowly but surely. We haven't delved into CSS too much, but you may have noticed that some elements have default styles we haven't declared within our CSS. That is the browser imposing its own preferred CSS styles for those elements. Fortunately we can overwrite these styles fairly easily, which is what we'll do next using CSS resets.

Using CSS Resets

Every web browser has its own default styles for different elements. How Google Chrome renders headings, paragraphs, lists, and so forth may be different from how Internet Explorer does. To ensure cross-browser compatibility, CSS resets have become widely used.

CSS resets take every common HTML element with a predefined style and provide one unified style for all browsers. These resets generally involve removing any sizing, margins, paddings, or additional styles and toning these values down. Because CSS cascades from top to bottom, our reset needs to be at the very top of our style sheet. Doing so ensures that those styles are read first and that all of the different web browsers are working from a common baseline.

There are a bunch of different resets available to use, all of which have their own fortes. One of the most popular resets is [Eric Meyer's reset](#), which has been adapted to include styles for the new HTML5 elements.

If you are feeling a bit more adventurous, there is also [Normalize.css](#), created by Nicolas Gallagher. Normalize.css focuses not on using a hard reset for all common elements, but instead on setting common styles for these elements. It requires a stronger understanding of CSS, as well as awareness of what you'd like your styles to be.

As previously mentioned, different browsers render elements in different ways. It's important to recognize the value in cross-browser compatibility and testing. Websites don't need to look exactly the same in every browser, but they should be close. Which browsers you wish to support, and to what degree, is a decision you will need to make based on what is best for your website.

In all there are a handful of things to be on the lookout for when writing CSS. The good news is that anything is possible, and with a little patience we'll figure it all out.

References

[Building Your First Web Page - Learn to Code HTML & CSS \(shayhowe.com\)](#)