

## How the Browser Reads CSS

People can read the same thing in completely different ways. One person reads it in one way, another person in another way, and a third one might even read the same thing in a totally third way. And the way they read it might even change if the context of where the written is written. And when it comes to the browser's way of reading CSS file(s), the above is exactly what is going on.


First of all, what does CSS stand for? CSS stands for **C**ascading **S**tyle **S**heet. The keyword here is "*Cascading*" since that's what it does. Styling given to a parent element will be inherited by all its children, since the styling cascades down onto the child(ren) element(s) from its/their parent.

You might ask, "Isn't CSS just CSS no matter what?". In terms of its purpose, which is to style a web document, yes. But then it also pretty much comes to an end there in terms of CSS just being CSS. There are different ways to write it, and the browser treats the different ways of writing CSS differently.

There are three ways to write CSS: *inline*, *internal*, and *external*. Let's break it down what those fancy terms even mean:

## 1. Inline

This means to style an element by writing the styling for it directly inside the opening tag of that element in the HTML file by using the style attribute. Let's say we want the text of the `<h1></h1>` to be red by styling it inline, we will do it like this, with the result of the text being red:



```
1. <h1 style="color: red;">
2.   I'm your fancy, big heading!
3. </h1>
```

But styling elements this way is very limited, and, therefore, something you should never do unless you have to test something styling-wise. Inline styling only styles the one element, on which it's applied, and not any other element of the same type. This means that if I had another `<h1></h1>`, it would **not** be red, since it has no styling to it.

## 2. Internal

This means to style the HTML inside `<style></style>` tags either right underneath the opening `<html>` tag or just above the closing `<body>` tag. This will work for really small applications but should never be used in applications planned to be put into production at some point.

If we were to style an `<h1></h1>` with internal styling, it would look like one of the following two examples with the first and the last example both having the result of printing out red text:

```
<!DOCTYPE html>
<head>
  <title>Styling an <h1></h1></title>
</head>

<html>
  <style>
    h1
    {
      color: red;
    }
  </style>
  <body>
    <h1>
      I'm your fancy, big heading!
    </h1>
  </body>
</html>
```

Example 1

```
<html>
  <body>
    <h1>
      I'm your fancy, big heading!
    </h1>
    <style>
      h1
      {
        color: red;
      }
    </style>
  </body>
</html>
```

Example 2

Where this differs from the inline styling is that this apply to any instance of the element, meaning that in our example, any `<h1></h1>` would be red.

### 3. External

This is undoubtedly the way to go. External means having your CSS in a separate file or multiple separate files that are all being linked to your HTML file(s). This is the way that's being used for production applications and is the best way of dealing with CSS and also the way you should do it.

So, how do we make an external CSS file, and how do we make it cooperate with the HTML file?

First of all, you have to create a CSS file. All CSS files have to have the .css extension at the end. That way, the file system on your machine knows it's dealing with a CSS file. You can call your file whatever you want (in the example below, it's called style.css and is not put in a folder but simply lies in the root of our project for simplicity purposes).

Thereafter, you have to have at least one HTML file, which has to end with the .html file extension at the end, so that your system knows it's dealing with an HTML file. Do note that if you only have one HTML file, it has to be named index.html (this makes the browser know that it's the root page).

In your HTML file, the CSS file has to be linked in the <head></head> tags. The way we link a CSS file to our HTML file is by using the <link> tag, in which we add a rel attribute of *stylesheet* and then the path to our CSS file. The path has to be exact. Otherwise, the browser is not going to find the CSS file.

The HTML and CSS should look like this with the result of red text when `run`:

```
1. <!DOCTYPE html>
2. <head>
3.   <link rel="stylesheet" href="style.css">
4. </head>
5. <html>
6. <body>
7.   <h1> I'm your fancy, big heading!</h1>
8. </body>
9. </html>
10.
11. <!-- style.css -->
12. h1
13. {
14.   color: red
15. }
```

Now you know the different ways available to write CSS, but that still doesn't answer the question being the topic of the post: How does the browser read CSS?

The different way of writing it has its own ranking in terms of precedence. The way of writing the CSS that the browser weighs the heaviest, is **inline**. This means that any styling that might have been added to the same element either internally or externally, will be overridden by the inline styling in terms of if the same property is written both inline or internally and/or externally.

L e t ' s   h a v e   a   l o o k   a t   a n   e x a m p l e :

```
<!DOCTYPE html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<html>
  <body>
    <div style="color: red;">
      Hey!
    </div>
  </body>
  <style>
    div
    {
      color: green;
    }
  </style>
</html>

<!-- style.css -->
div
{
  color: yellow;
}
```

If the example in the code is being **run**, the result would be that the color of the text turns out to be red.

The way of writing CSS that the browser weighs a little lesser, is internal. This means that any styling applied externally would be overridden by the internally added CSS (make sure that you do not have any inline styling applied, otherwise it's going to override the styling as described in the example above. Let's have a look at an example:



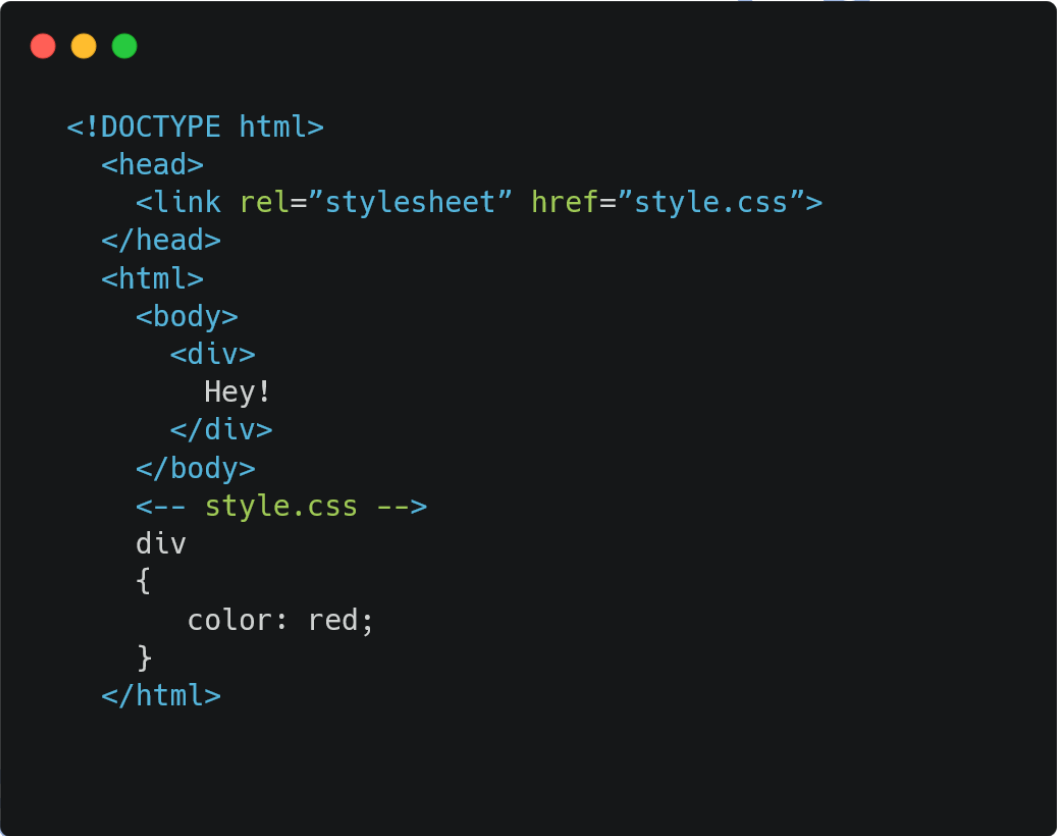
```
<!DOCTYPE html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<html>
<body>
  <div>
    Hey!
  </div>
</body>
<style>
  div
  {
    color: green;
  }
</style>

<-- style.css -->
div
{
  color: red;
}
</html>
```

Here, when the code is being run, the result will be that the text turns out to be green.

The way of writing CSS that the browser weighs the least, as you might have guessed by now, is external. You might have thought at first that it's the way the browser weighs the most, since that's the way you're used to write your CSS, but that's just not how things are. This means that the stylings written in the external CSS file are only to be applied, if no inline or internal stylings are being found by the browser. So, also here it's important to make sure that you do not have any inline or internal stylings. Otherwise, they'll override the stylings written in the external stylesheet.

Let's have a look at an example:



```
<!DOCTYPE html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<html>
  <body>
    <div>
      Hey!
    </div>
  </body>
  <-- style.css -->
  div
  {
    color: red;
  }
</html>
```

Here, when the code is being run, the result is that the text turns out to be red.



Now that you know how the browser reads CSS, it's your time to paint the picture you want inside the frame!

That's it!

- *Remember to make your website just a little bit more awesome every day.*

@WORLDWIDEWEBDEVELOPMENT