This lesson is going to be about writing **maintainable CSS code**. We will start with discovering the rules of selector efficiency and selector simplification. Then we're going to talk about the secret of descriptive CSS class names. This is a very interesting topic that's going to be very beneficial for you. At the end of this section, we're going to look at how to specify units in CSS in order to write maintainable code. Let's start with examining selector efficiency and selector simplification.

First, we're going to discover how selectors work in CSS, in other words, how browsers read and interpret selectors in CSS. Using this knowledge, we're going to optimize some CSS selectors.

All you need to understand about reading CSS selectors that they are read from right to left. This means that the browser first interprets the right-most selector and then moves leftwards. This is not too intuitive in case you write from left to right. However, this is exactly what you need to consider when optimizing your code.

The most efficient way to retrieve a selector is to find an identifier. This is because identifiers are unique in all your documents so you just need to find one identifier and then you're done.

The second most efficient way is classes. Even though identifiers may be slightly more efficient then classes, we will soon learn that I generally discourage you to use identifiers anywhere in your document. This is because identifiers are global resources and you cannot guarantee that in one part of your application you're not going to use an identifier that is already used somewhere else. For this reason, I highly recommend staying away from ID attributes in general. I know it might be slightly hard, for example, you might want to connect a label with an input field but there are other ways to do that.

The least efficient way of retrieving DOM nodes is using attributes. Imagine, instead of using the `input` tag if you just used `type=[text]`, then you would check every single DOM node if it has an attribute type, and if the type exists, whether it matches with text. It is very inefficient, so whenever you can, just stay away from retrieving attributes.

Don't take these rules very seriously though, because today's processors are very, very efficient.

There are many other mistakes that you can make that could cause problems of several magnitudes higher than just using attributes or classes. The differences might just be negligible. In fact, the way how you change selectors matters a lot more than the types of selectors that you use. Even if you use attributes selectors, you can lose a lot more time by chaining the wrong type of selectors after each other than by using the least efficient selector possible.

In extreme cases though, pay attention to selectors that might lead to retrieving thousands of DOM nodes. These could be retrieving all `div` of the page or the universal selector `*` which retrieves all DOM nodes.

Also, don't over qualify your selectors. For example, Don't write things like `li.header nav .item`. Over optimizing selector performance doesn't pay off. Focus your efforts on discovering other bottlenecks in your application and you're going to win a lot more in terms of performance. Focusing on selector efficiency too much is premature optimization which is a form of perfectionism and perfectionism rarely ever pays off in software engineering unless we have defined quality standards.

There is an option to improve your CSS efficiency with a very simple trick very quickly: use UnCSS tool. It removes unused CSS from your code automatically. You don't even have to think about it. You just have to execute it and you're going to make your code slightly more performant.