What is Less CSS and how to use Less CSS

Less CSS can make your programming language easier to use, with syntax shortcuts and powerful features. Here's what Less CSS can do for you.

Less CSS can make your programming language easier to use, with syntax shortcuts and powerful features. Here's **what Less CSS** can do for you.

If you are an experienced CSS programmer, you will understand the limitations of this language. It still lacks extended support for long-requested features like nesting and mixing.

Less (Leaner Style Sheets) is a CSS extension with many powerful features. If you know CSS, learning Less will be easier because the syntax of both is very similar.

How to install Less

You can install Less with the JavaScript package manager, NPM by running:

```
npm install less -g
```

Once installed, you can compile the **.less** file to **.css** using the lessc command. For example, the following command compiles **style.less** and outputs the results in a **style.css** file .

```
lessc style.less style.css
```

Variables in Less

Unlike normal CSS, the '--' operator is used to define variables. Less identifies variables with the '@' symbol. For example:

```
@width: 40px; @height: 80px;
```

This block of code simply creates two variables, width and height that hold two equivalent values: 40px and 80px. It usually takes common values ??in CSS and stores them in a variable. This makes it easier for the block of code to modify the above values ??since there is only one source of control.

Here's how you can use variables in Less. Create an **index.htm** file and add the following boilerplate code:

```
 Using Less CSS Hello from the children of planet Earth!
```

Next, create a style.less file and add:

```
@width: 400px; @height: 400px; @vertical-center: center; @txt-white: white; @bg-
```

Now you can compile the **.less** file to **.css** with the **lessc** command :

```
lessc style.less style.css
```

The compiled CSS will look like this:

```
div { width: 400px; height: 400px; display: flex; color: white; background-color
```

When you open your browser, this is what you'll see:

You can do more with variables in Less, for example interpolation allows you to use variables as selector names, URLs, etc. Here's an example of how variable interpolation is done:

```
@custom-selector: container; .@{custom-selector} { padding: 10px; margin: 10px; }
```

The above code block uses the @{.} clause to use a variable as a selector. When compiled, this code will produce the following output:

```
.container{ padding: 10px; margin: 10px; border: solid 10px; }
```

Arithmetic operations in Less

Less also provides support for arithmetic operations such as addition, subtraction, multiplication and division. These operations work with constants, values, and variables.

```
@variable-1: 5px; // Phép nhân gi?a bi?n và h?ng s?
 @variable-2: @variable-1 * 2 // Phép c?ng gi?a giá tr? và bi?
n @variable-3: 10px + @variable-2
```

How to use Mixins

Mixins allow you to reuse styles or CSS code via the stylesheet. Other CSS extensions like Sass also provide Mixins. To demonstrate how mixins work in Less, create index.htm and add the following code:

```
 Using Less CSS Lorem ipsum dolor sit amet, consectetur adipisicing elit. Soluta
```

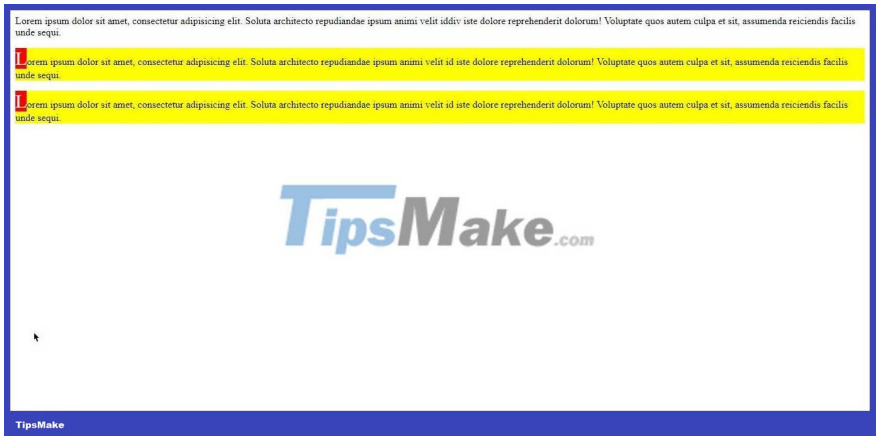
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Soluta architecto repudiandae ipsum animi velit id iste dolore reprehenderit dolorum! Voluptate quos autem culpa et sit, assumenda reiciendis facilis unde sequi.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Soluta architecto repudiandae ipsum animi velit id iste dolore reprehenderit dolorum! Voluptate quos autem culpa et sit, assumenda reiciendis facilis unde sequi.

Next, create a **style.less** file and add the following lines:

```
.sample-text() { background-color: yellow; } .first-letter() { background-color:
```

In the above code block, there are two mixin classes: . **sample-text** and **.first-letter** . When you want to use a mixin in another part of the style sheet, just reference it by name with parentheses at the end: **.mixin()** . In this browser you will see:



Styling Nesting in Less

Let's say you have a parent div with two child elements: a **p** element and another **div** . Normally, if you want to style the **p** element with red color and **the div** with green color, you can use the following method:

```
div p { color: red; } div { color: green }
```

Less provides similar functionality using **nesting** . In this case, the Less equivalent of the above code block is:

```
div { color: green; p { color: red; } }
```

This not only makes it easier to understand, but also more convenient to maintain the code. Referencing native selectors with less is easier than the **&** operator . For example:

```
button { background-color: blue; border: none; &:hover { background-color: grey;
```

The above code block will result in the following CSS code when compiled:

```
button { background-color: blue; border: none; } button:hover { background-color
```

Scopes and functions in Less

Like other common programming languages, variables have the scope of the block in which you define them. To illustrate this, create a new **index.htm** file , adding the first boilerplate HTML code originally provided. Then add the following block to the **body** tag :

```
 Outer Div should be red.
  Inner div should be green.
```

In the **style.less** file , add the following lines:

```
@bg-color: red; body { font-size: 40px; color: white; margin: 20px; } .inner-div
```

The inner-div block redefines the bg-color variable, which resides only within that block. Therefore, the green color only applies to that class and does not affect the global **bg-color variable.** When you compile and open the results in a browser, you'll see:

Less also provides handy functions that can be useful in some situations. For example, if you want to set a style only when a specific condition is met, you can do that using the if function. This function has the following syntax:

```
if((condition), value1, value2)
```

This code returns **value1** if the condition is met, otherwise it returns **value2** . For example:

```
@var1: 20px; @var2: 20px; div { padding: if((@var1 = @var2), 10px, 20px); }
```

The above code block will result in CSS after it is compiled:

```
div { padding: 10px; }
```

Features like functions, mixins, and variables are just a small part of what Less has to offer. Less is suitable for both small and large projects. Try and feel for yourself the wonderful things that Less brings to CSS.