**About the Tutorial**

LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for website. LESS is a dynamic style sheet language that extends the capability of CSS. LESS is also cross browser friendly.

**Audience**

This tutorial will help both students as well as professionals who want to make their websites or personal blogs more attractive.

**Prerequisites**

You should be familiar with:

- Basic word processing using any text editor.
- How to create directories and files.
- How to navigate through different directories.
- Internet browsing using popular browsers like Internet Explorer or Firefox.
- Developing simple webpages using HTML or XHTML.

If you are new to HTML and XHTML, then we suggest you go through our HTML Tutorial or XHTML Tutorial first.

**Copyright & Disclaimer**

© Copyright 2017 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com
Table of Contents

About the Tutorial...........................................................................................................i
Audience .........................................................................................................................i
Prerequisites ....................................................................................................................i
Copyright & Disclaimer .................................................................................................i
Table of Contents ...........................................................................................................ii

1. LESS — OVERVIEW ..................................................................................................1
2. LESS — INSTALLATION .........................................................................................3
3. LESS — NESTED RULES .......................................................................................9
4. LESS — NESTED DIRECTIVES & BUBBLING .....................................................12
5. LESS — OPERATIONS ...........................................................................................14
6. LESS — ESCAPING ................................................................................................16
7. LESS — FUNCTIONS ...............................................................................................18
8. LESS — NAMESPACES AND ACCESSORS ............................................................20
9. LESS — SCOPE .......................................................................................................22
10. LESS — COMMENTS ..............................................................................................24
11. LESS — IMPORTING ..............................................................................................26
12. LESS — VARIABLES ..............................................................................................29
    LESS – Variables Overview.....................................................................................29
    LESS – Variables Interpolation..............................................................................31
    LESS – Selectors ......................................................................................................32
    LESS – URLs ...........................................................................................................34
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15. LESS — PARAMETRIC MIXINS</td>
<td>88</td>
</tr>
<tr>
<td>- Mixins with Multiple Parameters</td>
<td>89</td>
</tr>
<tr>
<td>- Named Parameters</td>
<td>91</td>
</tr>
<tr>
<td>- @arguments Variable</td>
<td>93</td>
</tr>
<tr>
<td>- Less Advanced Arguments and @rest Variable</td>
<td>94</td>
</tr>
<tr>
<td>- Pattern Matching</td>
<td>95</td>
</tr>
<tr>
<td>16. LESS — MIXINS AS FUNCTIONS</td>
<td>98</td>
</tr>
<tr>
<td>- Mixin scope</td>
<td>98</td>
</tr>
<tr>
<td>- Mixin scope</td>
<td>100</td>
</tr>
<tr>
<td>- Mixin &amp; return values</td>
<td>102</td>
</tr>
<tr>
<td>- Mixin inside mixin</td>
<td>103</td>
</tr>
<tr>
<td>17. LESS — PASSING RULESETS TO MIXINS</td>
<td>106</td>
</tr>
<tr>
<td>- Scoping</td>
<td>107</td>
</tr>
<tr>
<td>- Definition and Caller Scope Visibility</td>
<td>108</td>
</tr>
<tr>
<td>- Referencing Won’t Modify Detached Ruleset Scope</td>
<td>110</td>
</tr>
<tr>
<td>- Unlocking Will Modify Detached Ruleset Scope</td>
<td>111</td>
</tr>
<tr>
<td>18. LESS — IMPORT DIRECTIVES</td>
<td>114</td>
</tr>
<tr>
<td>- File Extensions</td>
<td>114</td>
</tr>
<tr>
<td>19. LESS — IMPORT OPTIONS</td>
<td>117</td>
</tr>
<tr>
<td>- Import Options Reference Keyword</td>
<td>117</td>
</tr>
<tr>
<td>- Import Options Extend</td>
<td>120</td>
</tr>
<tr>
<td>- Import Options Inline Keyword</td>
<td>122</td>
</tr>
<tr>
<td>- Less Import Options Less Keyword</td>
<td>124</td>
</tr>
<tr>
<td>- Import Options CSS Keyword</td>
<td>126</td>
</tr>
<tr>
<td>- Import Options Once Keyword</td>
<td>128</td>
</tr>
</tbody>
</table>
LESS - Unit Function.................................................................................................................. 176
Less - get - unit Function.......................................................................................................... 178
Less - svg gradient Function.................................................................................................... 180

26. LESS — STRING FUNCTIONS .......................................................................................... 182

27. LESS — LIST FUNCTIONS ............................................................................................... 184
    LESS - List Length Function............................................................................................... 184
    LESS - List Extract Function ............................................................................................ 185

28. LESS — MATH FUNCTIONS ............................................................................................. 188

29. LESS — TYPE FUNCTIONS .............................................................................................. 192

30. LESS — COLOR DEFINATION FUNCTIONS ..................................................................... 195

31. LESS — COLOR CHANNEL FUNCTIONS ......................................................................... 198

32. LESS — COLOR OPERATION ............................................................................................ 201
    Less - Saturate ...................................................................................................................... 202
    Less - Desaturate ................................................................................................................. 205
    Less - Lighten ..................................................................................................................... 207
    Less - Darken ...................................................................................................................... 210
    Less - Fadein ...................................................................................................................... 213
    Less - Fadeout .................................................................................................................... 217
    Less - Fade ......................................................................................................................... 220
    Less - Spin .......................................................................................................................... 223
    Less - Mix ............................................................................................................................ 226
    Less - Tint ........................................................................................................................... 228
    Less - Shade ....................................................................................................................... 231
    Less - Greyscale ............................................................................................................... 234
LESS – Contrast ........................................................................................................................................236

33. LESS – COLOR BLENDING FUNCTIONS .........................................................................................240

Less – Color Blending Multiply Function ..........................................................................................241
Less – Color Blending Screen Function ............................................................................................244
Less – Color Blending Overlay Function ..........................................................................................248
Less – Color Blending Softlight Function ..........................................................................................251
Less – Color Blending Hardlight Function .......................................................................................255
Less – Color Blending Difference Function .....................................................................................258
Less – Color Blending Exclusion Function .......................................................................................261
Less – Color Blending Average Function ..........................................................................................264
Less – Color Blending Negation Function ..........................................................................................267

34. LESS – COMMAND LINE USAGE .................................................................................................271

35. LESS – USING LESS IN THE BROWSER .......................................................................................276

36. LESS — BROWSER SUPPORT .....................................................................................................280

37. LESS – PLUGINS .............................................................................................................................281

38. LESS – PROGRAMMATIC USAGE ................................................................................................285

39. LESS – ONLINE COMPILERS .......................................................................................................286

40. LESS – GUIS ...................................................................................................................................287

41. LESS – EDITORS AND PLUGINS ..................................................................................................289

  Sublime Text 2 & 3 ............................................................................................................................290
  Eclipse ...............................................................................................................................................291
  Visual Studio ....................................................................................................................................291
  Dreamweaver ....................................................................................................................................291
  Notepad++ 6.x ...................................................................................................................................292
42. LESS – THIRD PARTY COMPILERS ...................................................................... 293
    Node.js Compilers ................................................................................................. 293
    Other Technologies .................................................................................................. 293

43. LESS – FRAMEWORKS ......................................................................................... 295
    UI/Theme Frameworks and Components ................................................................ 295
    Grid Systems ............................................................................................................ 297
    Mixin Libraries ........................................................................................................ 297
LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for website. LESS is a dynamic style sheet language that extends the capability of CSS. LESS is also cross browser friendly.

CSS Preprocessor is a scripting language that extends CSS and gets compiled into regular CSS syntax, so that it can be read by your web browser. It provides functionalities like variables, functions, mixins and operations that allow you to build dynamic CSS.

Why LESS?
Let us now understand why do we use LESS.

- LESS supports creating cleaner, cross-browser friendly CSS faster and easier.
- LESS is designed in JavaScript and also created to be used in live, which compiles faster than other CSS pre-processors.
- LESS keeps your code in modular way which is really important by making it readable and easily changeable.
- Faster maintenance can be achieved by the use of LESS variables.

History
LESS was designed by Alexis Sellier in 2009. LESS is an open-source. The first version of LESS was written in Ruby; in the later versions, the use of Ruby was replaced by JavaScript.

Features
- Cleaner and more readable code can be written in an organized way.
- We can define styles and it can be reused throughout the code.
- LESS is based on JavaScript and is a super set of CSS.
- LESS is an agile tool that sorts out the problem of code redundancy.

Advantages
- LESS easily generates CSS that works across the browsers.
- LESS enables you to write better and well-organized code by using nesting.
- Maintenance can be achieved faster by the use of variables.
• LESS enables you to reuse the whole classes easily by referencing them in your rule sets.
• LESS provides the use of operations that makes coding faster and saves time.

Disadvantages
• It takes time to learn if you are new to CSS preprocessing.
• Due to the tight coupling between the modules, more efforts should be taken to reuse and/or test dependent modules.
• LESS has less framework compared to older preprocessor like SASS, which consists of frameworks Compass, Gravity and Susy.
In this chapter, we will understand, in a step-by-step manner, how to install LESS.

System Requirements for LESS

- **Operating System:** Cross-platform
- **Browser Support:** IE (Internet Explorer 8+), Firefox, Google Chrome, Safari.

Installation of LESS

Let us now understand the installation of LESS.

**Step 1:** We need **NodeJs** to run LESS examples. To download NodeJs, open the link [https://nodejs.org/en/](https://nodejs.org/en/), you will see a screen as shown below:

![Node.js download page](image)

Download the *Latest Features* version of the zip file.

**Step 2:** Run the setup to install the **Node.js** on your system.

**Step 3:** Install LESS on the server via NPM (Node Package Manager). Run the following command in the command prompt.
Step 4: After successful installation of LESS, you will see the following lines on the command prompt –

```
`-- less@2.6.1  
  `-- errno@0.1.4  
    `-- prr@0.0.0  
  `-- graceful-fs@4.1.3  
  `-- image-size@0.4.0  
  `-- mime@1.3.4  
  `-- mkdirp@0.5.1  
    `-- asap@2.0.3  
  `-- request@2.69.0  
    `-- aws-sign2@0.6.0  
      `-- aws4@1.3.2  
       `-- lru-cache@4.0.0  
       `-- pseudomap@1.0.2  
        `-- yallist@2.0.0  
      `-- bl@1.0.3  
        `-- readable-stream@2.0.6  
          `-- core-util-is@1.0.2  
            `-- inherits@2.0.1  
            `-- isarray@1.0.0  
            `-- process-nextick-args@1.0.6  
          `-- string_decoder@0.10.31  
            `-- util-deprecate@1.0.2  
            `-- caseless@0.11.0  
          `-- combined-stream@1.0.5  
            `-- delayed-stream@1.0.0
```
Example
Following is a simple example of LESS.

hello.htm

```html
<!doctype html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
  <h1>Welcome to TutorialsPoint</h1>
  <h3>Hello!!!!!</h3>
</body>
```
Let us now create a file `style.less` which is quite similar to CSS, the only difference is that it will be saved with `.less` extension. Both the files, `.html` and `.less` should be created inside the folder `nodejs`.

**style.less**

```less
@primarycolor: #FF7F50;
@color: #800080;

h1{
  color: @primarycolor;
}

h3{
  color: @color;
}
```

Compile `style.less` file to `style.css` by using the following command:

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

When you run the above command, it will create the `style.css` file automatically. Whenever you change the LESS file, it's necessary to run the above command in the `cmd` and then the `style.css` file will get updated.

The `style.css` file will have the following code when you run the above command:

**style.css**

```css
h1 {
  color: #FF7F50;
}
```
Output

Let us now carry out the following steps to see how the above code works:

- Save the above html code in `hello.htm` file.
- Open this HTML file in a browser, the following output will get displayed.

![Welcome to TutorialsPoint](image)
Language Features
3. LESS — NESTED RULES

Description
It is a group of CSS properties which allows using properties of one class into another class and includes the class name as its properties. In LESS, you can declare mixin in the same way as CSS style using class or id selector. It can store multiple values and can be reused in the code whenever necessary.

Example
The following example demonstrates the use of nested rules in the LESS file:

```html
<html>
<head>
    <title>Nested Rules</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
    <div class="container">
        <h1>First Heading</h1>
        <p>LESS is a dynamic style sheet language that extends the capability of CSS.</p>
        <div class="myclass">
            <h1>Second Heading</h1>
            <p>LESS enables customizable, manageable and reusable style sheet for website.</p>
        </div>
    </div>
</body>
</html>
```

Next, create the style.less file.

style.less

```css
.container{
```
You can compile the *style.less* file to *style.css* by using the following command:

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

Execute the above command, it will create the *style.css* file automatically with the following code:

```
.style.css

.container h1 {
    font-size: 25px;
    color: #E45456;
}
```

19
Output

Follow these steps to see how the above code works:

- Save the above html code in nested_rules.html file.
- Open this HTML file in a browser, the following output gets displayed.

```html
.container p {
    font-size: 25px;
    color: #3C7949;
}
.container .myclass h1 {
    font-size: 25px;
    color: #E45456;
}
.container .myclass p {
    font-size: 25px;
    color: #3C7949;
}
```

**First Heading**

LESS is a dynamic style sheet language that extends the capability of CSS.

**Second Heading**

LESS enables customizable, manageable and reusable style sheet for web site.
4. LESS — NESTED DIRECTIVES & BUBBLING

Description
You can nest the directives such as media and keyframe in the same manner, the way you nest the selectors. You can place the directive on top and its relative elements will not be changed inside its rule set. This is known as the bubbling process.

Example
The following example demonstrates the use of the nested directives and bubbling in the LESS file:

```html
<html>
<head>
    <title>Nested Directives</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
    <h1>Example using Nested Directives</h1>
    <p class="myclass">LESS enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the file `style.less`.

style.less

```css
.myclass {
    @media screen {
        color: blue;
        @media (min-width: 1024px) {
            color: green;
        }
    }
}
```
You can compile the *style.less* file to *style.css* by using the following command:

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

Execute the above command, it will create the *style.css* file automatically with the following code:

```
@media screen {
  .myclass {
    color: blue;
  }
}
@media screen and (min-width: 1024px) {
  .myclass {
    color: green;
  }
}
@media mytext {
  .myclass {
    color: black;
  }
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in *nested_directives_bubbling.html* file.
- Open this HTML file in a browser, the following output will get displayed.
Example using Nested Directives

LESS enables customizable, manageable and reusable style sheet for web site.
Description
LESS provides support for some arithmetical operations such as plus (+), minus (-), multiplication (*) and division (/) and they can operate on any number, color or variable. Operations save lot of time when you are using variables and you feel like working on simple mathematics.

Example
The following example demonstrates the use of operations in the LESS file:

```html
<html>
<head>
  <title>Less Operations</title>
  <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
  <h1>Example using Operations</h1>
  <p class="myclass">LESS enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the file `style.less`.

**style.less**

```less
@fontSize: 10px;
.myclass {
  font-size: @fontSize * 2;
  color:green;
}
```

You can compile the `style.less` file to `style.css` by using the following command:
Execute the above command, it will create the `style.css` file automatically with the following code:

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

**style.css**

```css
.myclass {
    font-size: 20px;
    color: green;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in `operations.html` file.
- Open this HTML file in a browser, the following output will get displayed.
6. LESS — ESCAPING

Description
It builds selectors dynamically and uses property or variable value as arbitrary string.

Example
The following example demonstrates the use of escaping in the LESS file:

```html
<html>
<head>
  <title>Less Escaping</title>
  <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
  <h1>Example using Escaping</h1>
  <p>LESS enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the file style.less.

**style.less**

```less
p {
  color: ~"green";
}
```

You can compile the style.less file to style.css by using the following command:

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

Execute the above command, it will create the style.css file automatically with the following code:
**style.css**

```css
p {
    color: green;
}
```

Anything written inside "some_text" will be displayed as `some_text` after compiling the LESS code to CSS code.

**Output**

Let us now perform the following steps to see how the above code works:

- Save the above html code in **escaping.html** file.
- Open this HTML file in a browser, the following output will get displayed.

![Example using Escaping](file://C:/nodejs/escaping.html)
7. LESS — FUNCTIONS

Description
LESS maps JavaScript code with manipulation of values and uses predefined functions to manipulate HTML elements aspects in the style sheet. It provides several functions to manipulate colors such as round function, floor function, ceil function, percentage function, etc.

Example
The following example demonstrates the use of functions in the LESS file:

```html
<html>
<head>
  <title>Less Functions</title>
  <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
  <h1>Example using Functions</h1>
  <p class="mycolor">LESS enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the file style.less.

**style.less**

```less
@color: #FF8000;
@width:1.0;
.mycolor{
  color: @color;
  width: percentage(@width);
}
```
You can compile the `style.less` file to `style.css` by using the following command:

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

Now execute the above command; it will create the `style.css` file automatically with the following code:

```
.style.css

.mycolor {
  color: #FF8000;
  width: 100%;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in `functions.html` file.
- Open this HTML file in a browser, you will receive the following output.

```
Example using Functions

LESS enables customizable, manageable and reusable style sheet for web site.
```
**Description**

Namespaces are used to group the mixins under a common name. Using namespaces, you can avoid conflict in name and encapsulate a group of mixins from outside.

**Example**

The following example demonstrates the use of namespaces and accessors in the LESS file:

```html
<html>
<head>
    <title>Less Namespaces and Accessors</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
    <h1>Example using Namespaces and Accessors</h1>
    <p class="myclass">LESS enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the file *style.less*.

**style.less**

```less
.class1 {
    .class2 {
        .val(@param) {
            font-size: @param;
            color:green;
        }
    }
}
```

...
You can compile the *style.less* file to *style.css* by using the following command:

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

Execute the above command; it will create the *style.css* file automatically with the following code:

```
.style.css
.myclass {
  font-size: 20px;
  color: green;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in *namespaces_accessors.html* file.
- Open this HTML file in a browser, the following output gets displayed.

![Example using Namespaces and Accessors](file:///C:/nodejs/namespaces_accessors.html)

*LESS enables customizable, manageable and reusable style sheet for web site.*
Description
Variable scope specifies the place of the available variable. The variables will be searched from the local scope and if they are not available, then compiler will search from the parent scope.

Example
The following example demonstrates the use of namespaces and accessors in the LESS file:

```html
<html>
<head>
    <title>Less Scope</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
    <h1>Example using Scope</h1>
    <p class="myclass">LESS enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the file `style.less`.

**style.less**

```less
@var: @a;
@a: 15px;

.myclass {
    font-size: @var;
    @a:20px;
    color: green;}
```
You can compile the `style.less` file to `style.css` by using the following command:

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

Execute the above command; it will create the `style.css` file automatically with the following code:

```
.style.css

.myclass {
   font-size: 20px;
   color: green;
}
```

### Output

Follow these steps to see how the above code works:

- Save the above html code in `scope.html` file.
- Open this HTML file in a browser, the following output gets displayed.

![Example using Scope](https://via.placeholder.com/150)
Description
Comments make the code clear and understandable for the users. You can use both the block style and the inline comments in the code, but when you compile the LESS code, the single line comments will not appear in the CSS file.

Example
The following example demonstrates the use of comments in the LESS file:

```html
<html>
<head>
    <title>Less Comments</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
    <h1>Example using Comments</h1>
    <p class="myclass">LESS enables customizable, manageable and reusable style sheet for web site.</p>
    <p class="myclass1">It allows reusing CSS code and writing LESS code with same semantics.</p>
</body>
</html>
```

Next, create the file `style.less`.

**style.less**

```less
/* It displays the green color! */
.myclass{
    color: green;
}
// It displays the blue color
```


You can compile the `style.less` file to `style.css` by using the following command:

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

Now execute the above command, it will create the `style.css` file automatically with the following code:

```
/* It displays the green color! */
.style.css
.myclass {
  color: green;
}
.myclass1 {
  color: red;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in `comments.html` file.
- Open this HTML file in a browser, the following output gets displayed.
Example using Comments

LESS enables customizable, manageable and reusable style sheet for web site.

It allows reusing CSS code and writing LESS code with same semantics.
11. LESS — IMPORTING

Description
It is used to import the contents of the LESS or CSS files.

Example
The following example demonstrates the use of importing in the LESS file:

```html
<html>
<head>
    <title>Less Importing</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
    <h1>Example using Importing</h1>
    <p class="myclass">LESS enables customizable, manageable and reusable style sheet for web site.</p>
    <p class="myclass1">It allows reusing CSS code and writing LESS code with same semantics.</p>
    <p class="myclass2">LESS supports creating cleaner, cross-browser friendly CSS faster and easier.</p>
</body>
</html>
```

Next, create the file `myfile.less`.

**myfile.less**

```less
.myclass{
    color: #FF8000;
}
.myclass1{
    color: #5882FA;
}
```
Now create the style.less file.

**style.less**

```less
@import "http://www.tutorialspoint.com/less/myfile.less";

.myclass2
{
  color: #FF0000;
}
```

The `myfile.less` file which will be imported into `style.less` from the path [http://www.tutorialspoint.com/less/myfile.less](http://www.tutorialspoint.com/less/myfile.less)

You can compile the `style.less` file to `style.css` by using the following command:

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

Execute the above command, it will create the `style.css` file automatically with the following code:

**style.css**

```css
.myclass {
  color: #FF8000;
}
.myclass1 {
  color: #5882FA;
}
.myclass2 {
  color: #FF0000;
}
```
Output

Follow these steps to see how the above code works:

- Save the above html code in the `importing.html` file.
- Open this HTML file in a browser, the following output gets displayed.

![Example using Importing](image)

LESS enables customizable, manageable and reusable style sheet for web site.

It allows reusing CSS code and writing LESS code with same semantics.

LESS supports creating cleaner, cross-browser friendly CSS faster and easier.
In this chapter, we will discuss the Variables in LESS. LESS allows variables to be defined with an @ symbol. The Variable assignment is done with a colon (:).

The following table demonstrates the use of LESS variables in detail.

<table>
<thead>
<tr>
<th>S.NO.</th>
<th>Variables usage &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Overview</strong>&lt;br&gt;Repetition of same value many times can be avoided by the use of variables.</td>
</tr>
<tr>
<td>2</td>
<td><strong>Variable Interpolation</strong>&lt;br&gt;The variables can also be used in other places like selector names, property names, URLs and @import statements.</td>
</tr>
<tr>
<td>3</td>
<td><strong>Variable Names</strong>&lt;br&gt;We can define variables with a variable name consisting of a value.</td>
</tr>
<tr>
<td>4</td>
<td><strong>Lazy Loading</strong>&lt;br&gt;In lazy loading, variables can be used even when they are not.</td>
</tr>
<tr>
<td>5</td>
<td><strong>Default Variables</strong>&lt;br&gt;Default variable has an ability to set a variable only when it is not already set. This feature is not required because variables can be easily overridden by defining them afterwards.</td>
</tr>
</tbody>
</table>

**LESS — Variables Overview**

**Description**

Repetition of the same value many times is usually seen across your stylesheet. Instead of using the same value multiple times, variables can be used. It makes maintenance of code easier and those values can be controlled from single location.

**Example**
The following example demonstrates the use of variables in the LESS file:

```html
<html>
<head>
    <link rel="stylesheet" href="style.css" type="text/css" />
    <title>LESS variables overview</title>
</head>
<body>
    <h1>Welcome to Tutorialspoint</h1>
    <div class="div1">
        <p>LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
    </div>
    <div class="div2">
        <p>LESS is a dynamic style sheet language that extends the capability of CSS. LESS is also cross browser friendly.</p>
    </div>
</body>
</html>
```

Next, create the file `style.less`.

**style.less**

```less
@color1: #ca428b;
.div1{
    background-color : @color1;
}
.div2{
    background-color : @color1;
}
```

You can compile the `style.less` to `style.css` by using the following command:

```
lessc style.less style.css
```
Execute the above command, it will create the style.css file automatically with the following code:

```less
style.css

h1 {
  color: #D0DC11;
}
.div1 {
  background-color: #ca428b;
  color: #D0DC11;
}
.div2 {
  background-color: #ca428b;
  color: #D0DC11;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in `less_variables_overview.html` file.
- Open this HTML file in a browser, the following output gets displayed.
LESS — Variables Interpolation

Description
The variable interpolation is the process of evaluating an expression or literal containing one or more variables, yielding output in which the variables are replaced with their corresponding values. The variables can also be used in other places like selector names, property names, URLs and @import statements.
The following table demonstrates the use of *variable interpolation* in detail.

<table>
<thead>
<tr>
<th>S.NO.</th>
<th>Variables usage &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Selectors</td>
</tr>
<tr>
<td></td>
<td>The selector can reference any variable and it is built during the compile time.</td>
</tr>
<tr>
<td>2</td>
<td>URLs</td>
</tr>
<tr>
<td></td>
<td>The variables can be used to hold URLs.</td>
</tr>
<tr>
<td>3</td>
<td>Import Statements</td>
</tr>
<tr>
<td></td>
<td>An import statement can have a variable which holds a path.</td>
</tr>
<tr>
<td>4</td>
<td>Properties</td>
</tr>
<tr>
<td></td>
<td>The variables can be referenced by properties.</td>
</tr>
</tbody>
</table>

**LESS — Selectors**

**Description**

The selector can reference any variable and it is built during the compile time. The variable name must be placed inside the curly braces ({} ) prefixed with the @ symbol.

**Example**

The following example demonstrates the use of selector in the LESS file:

```html
<html>
  <head>
    <link rel = "stylesheet" href = "style.css" type = "text/css" />
    <title>LESS selectors</title>
  </head>

  <body>
    <h2>Welcome to Tutorialspoint</h2>
    <div class = "div1">
      <p>LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
    </div>
  </body>
</html>
```
LESS is a dynamic style sheet language that extends the capability of CSS. LESS is also cross browser friendly.

Next, create the file `style.less`.

```less
@selector: h2;

@{selector} {
    background: #2ECCFA;
}
```

You can compile the `style.less` to `style.css` by using the following command:

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

Execute the above command, it will create the `style.css` file automatically with the following code:

```css
h2 {
    background: #2ECCFA;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in the `less_variables_interpolation_selectors.html` file.
- Open this HTML file in a browser, the following output gets displayed.
LESS — URLs

Description
The variables can be used to hold the URLs.

Example
The following example demonstrates the use of variables to hold URL in the LESS file:

```html
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>LESS URLs</title>
</head>
<body>
<div class="myclass">
</div>
</body>
</html>
```

Next, create the file `style.less`.

**style.less**

```less
@images: "http://www.tutorialspoint.com";

.myclass {
  background: url("@{images}/less/images/less_variables/birds.jpg");
  width:800px;
}
```
You can compile the `style.less` to `style.css` by using the following command:

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

Execute the above command, it will create the `style.css` file automatically with the following code:

```
.style.css

.myclass {
  background: url("http://www.tutorialspoint.com/less/images/less_variables/birds.jpg");
  width: 800px;
  height: 500px;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in `less_variables_interpolation_url.html` file.
- Open this HTML file in a browser, the following output gets displayed.
LESS — Import Statements

Description
An import statement can have a variable which holds a path. This is very useful when you are referring a common parent directory.

Example
The following example demonstrates the use of variables in the import statement:

```html
<html>
<head>
    <link rel="stylesheet" href="style.css" type="text/css" />
    <title>LESS Variables in Import Statements</title>
</head>
<body>
<div class="myclass">
    <h2>Welcome to Tutorialspoint</h2>
    <p>LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
</div>
</body>
</html>
```

Next, create the style.less file.

**style.less**

```less
@path : "http://www.tutorialspoint.com/less";
@import "@{path}/external1.less";
.myclass{
    color : #A52A2A;
}
```

The following code will import the external.less file into style.less from the http://www.tutorialspoint.com/less/external1.less path:

**external1.less**
You can compile the `style.less` to `style.css` by using the following command:

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

Execute the above command, it will create the `style.css` file automatically with the following code:

**style.css**

```css
body {
  background: #C0C0C0;
}
p {
  color: #A52A2A;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in `less_variables_interpolation_import.html` file.
- Open this HTML file in a browser, the following output gets displayed.
LESS — Variables Interpolation Properties

Description
The variables can be referenced by properties.

Example
The following example demonstrates the use of variables referenced by properties in the LESS files:

```html
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>LESS Variables Interpolation Properties</title>
</head>
<body>
<div class="myclass">
  <h2>Welcome to Tutorialspoint</h2>
  <p>LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
</div>
</body>
</html>
```

Next, create the file `style.less`.

```less
@my-property: color;
.myclass {
  background-@{my-property}: #81F7D8;
}
```

You can compile the `style.less` to `style.css` by using the following command:

```
lessc style.less style.css
```
Execute the above command, it will create the style.css file automatically with the following code:

**style.css**

```css
=myclass {
  background-color: #81F7D8;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in `less_variables_interpolation_properties.html` file.
- Open this HTML file in a browser, the following output gets displayed.

![Image showing the output of the code](image)

**LESS — Variable Names**

**Description**

We can define the variables with a variable name consisting of a value.

**Example**

The following example demonstrates the use of `variable` holding another `variable` in the LESS file:

```html
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
```
Next, create the file `style.less`.

```
.style.less

.myclass{
  @col: #ca428b;
  @color: "col";
  background-color: @color;
}
```

You can compile the `style.less` to `style.css` by using the following command:

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

Execute the above command, it will create the `style.css` file automatically with the following code:

```
.style.css

myclass {
  background-color: #ca428b;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in `less_variables_names.html` file.
• Open this HTML file in a browser, the following output gets displayed.

**Welcome to TutorialsPoint**

LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.

---

**LESS — Variable Lazy Loading**

**Description**

In lazy loading, variables can be used even when they are not declared.

**Example**

The following example demonstrates the use of *lazy loading of variable* in the LESS file:

```html
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>LESS Lazy Loading</title>
</head>
<body>
<h2>Welcome to Tutorialspoint</h2>
<p>LESS is a CSS pre-processor.</p>
</body>
</html>
```

Next, create the file *style.less*.

**style.less**

```less
p {
  font-size: @a;
  color: #ca428b;
}
```
@a: @b;
@a: 25px;

You can compile the style.less to style.css by using the following command:

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

Execute the above command, it will create the style.css file automatically with the following code:

**style.css**

```
p {
    font-size: 25px;
    color: #ca428b;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in **less_lazy_loading.html** file.
- Open this HTML file in a browser, the following output gets displayed.

If you define a variable twice, the last definition of the variable from the current scope is searched and used. For more details click here.
LESS — Variable Lazy Loading Scope

Description
If you define a variable two times, the last definition of the variable from the current scope is searched and used. This method is similar to CSS itself where the value is extracted from the last property inside a definition.

Example
The following example demonstrates the use of lazy loading of variable in a different scope in the LESS file:

```html
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>LESS Lazy Loading in Different Scope</title>
</head>
<body>
<div class="myclass">
  <p>Welcome to Tutorialspoint</p>
  <p class="para1">LESS is a CSS pre-processor.</p>
</div>
</body>
</html>
```

Next, create the file style.less.

**style.less**

```less
@var: 10;
.myclass {
  @var: 50;
  .para1 {
    @var: 30;
    font-size: @var;
    @var: 20;
  }
```

56
You can compile the `style.less` to `style.css` by using the following command:

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

Execute the above command, it will create the `style.css` file automatically with the following code:

```
.style.css

.myclass {
  font-size: 50;
}

.myclass .para1 {
  font-size: 20;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in `less_lazy_loading_scope.html` file.
- Open this HTML file in a browser, the following output gets displayed.
**LESS — Default Variables**

**Description**
Default variable has an ability to set a variable only when it's not already set. This feature is not required because variables can be easily overridden by defining them afterwards.

**Example**
The following example demonstrates the use of *default variables* in the LESS file:

```html
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>LESS Default Variables</title>
</head>
<body>
<h1>Welcome to TutorialsPoint</h1>
<p>LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the file *style.less*.

**style.less**

```less
@import "http://www.tutorialspoint.com/less/lib.less"; // first declaration of 
@color
@color: green; // this will override @color defined previously
p{
  color : @color;
}
```

The following code imports the *lib.less* file into *style.less* from the [http://www.tutorialspoint.com/less/lib.less](http://www.tutorialspoint.com/less/lib.less) path:

**lib.less**

---

**tutorialspoint**

SIMPLY EASY LEARNING

58
@color: blue;

You can compile the style.less to style.css by using the following command:

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

Execute the above command, it will create the style.css file automatically with the following code:

**style.css**

```css
p {
  color: green;
}
```

**Output**

Follow these steps to see how the above code works:

- Save the above html code in `less_default_variables.html` file.
- Open this HTML file in a browser, the following output gets displayed.
End of ebook preview
If you liked what you saw...
Buy it from our store @ https://store.tutorialspoint.com