{less}

tutorialspoint

SIMPLYEASYLEARNING

{less}

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

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

Table of Contents

1. LESS — Overview

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*.

2. LESS — Installation

In this chapter, we will understand, in a step-by-step manner, how to install LESS.

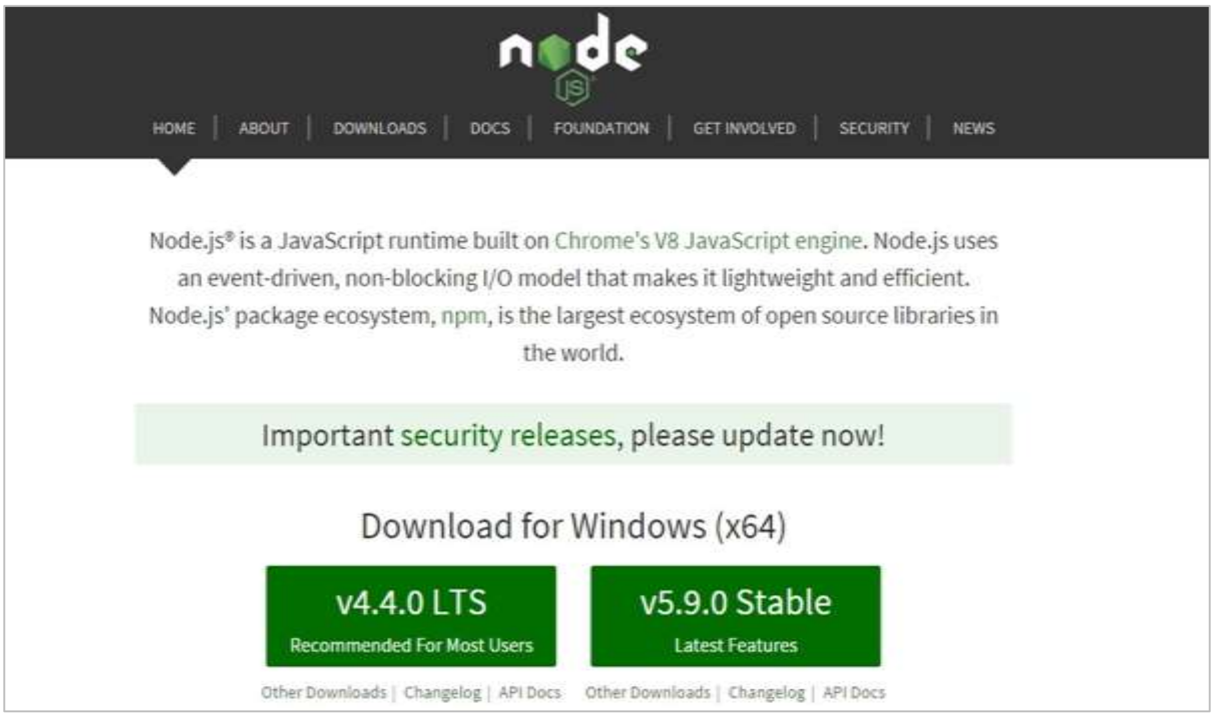
System Requirements for LESS

- **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/, you will see a screen as shown below:


node JS
HOME | ABOUT | DOWNLOADS | DOCS | FOUNDATION | GET INVOLVED | SECURITY | NEWS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Important security releases, please update now!

Download for Windows (x64)

v4.4.0 LTS
Recommended For Most Users
v5.9.0 Stable
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs

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.

```
npm install -g less
```

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
  | `-- minimist@0.0.8
  +-- promise@7.1.1
  | `-- 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
  | +-- extend@3.0.0
  | +-- forever-agent@0.6.1
  | +-- form-data@1.0.0-rc4
  | | `-- async@1.5.2
  | +-- har-validator@2.0.6
  | | +-- chalk@1.1.1
  | | | +-- ansi-styles@2.2.0
  | | | | `-- color-convert@1.0.0
  | | | +-- escape-string-regexp@1.0.5
  | | | +-- has-ansi@2.0.0
  | | | | `-- ansi-regex@2.0.0
  | | | +-- strip-ansi@3.0.1
  | | | `-- supports-color@2.0.0
  | | +-- commander@2.9.0
  | | | `-- graceful-readlink@1.0.1
  | | +-- is-my-json-valid@2.13.1
  | | | +-- generate-function@2.0.0
  | | | +-- generate-object-property@1.2.0
  | | | | `-- is-property@1.0.2
  | | | +-- jsonpointer@2.0.0
  | | | `-- xtend@4.0.1
  | | `-- pinkie-promise@2.0.0
  | |   `-- pinkie@2.0.4
  | +-- hawk@3.1.3
  | | +-- boom@2.10.1
  | | +-- cryptiles@2.0.5
  | | +-- hoek@2.16.3
  | | `-- sntp@1.0.9
  | +-- http-signature@1.1.1
  | | +-- assert-plus@0.2.0
  | | +-- jsprim@1.2.2
  | | | +-- extsprintf@1.0.2
  | | | +-- json-schema@0.2.2
  | | | `-- verror@1.3.6
  | | `-- sshpk@1.7.4
  | |   +-- asn1@0.2.3
  | |   +-- dashdash@1.13.0
  | |   | `-- assert-plus@1.0.0
  | |   +-- ecc-jsbn@0.1.1
  | |   +-- jodid25519@1.0.2
  | |   +-- jsbn@0.1.0
  | |   `-- tweetnacl@0.14.1
  | +-- is-typedarray@1.0.0
  | +-- isstream@0.1.2
  | +-- json-stringify-safe@5.0.1
```

```
  | +-- mime-types@2.1.10
  | | `-- mime-db@1.22.0
  | +-- node-uuid@1.4.7
  | +-- oauth-sign@0.8.1
  | +-- qs@6.0.2
  | +-- stringstream@0.0.5
  | +-- tough-cookie@2.2.2
  | `-- tunnel-agent@0.4.2
  `-- source-map@0.5.3
```

Example

Following is a simple example of LESS.

hello.htm

```
<!doctype html>
<head>
     <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
     <h1>Welcome to TutorialsPoint</h1>
     <h3>Hello!!!!!</h3>
</body>
</html>
```

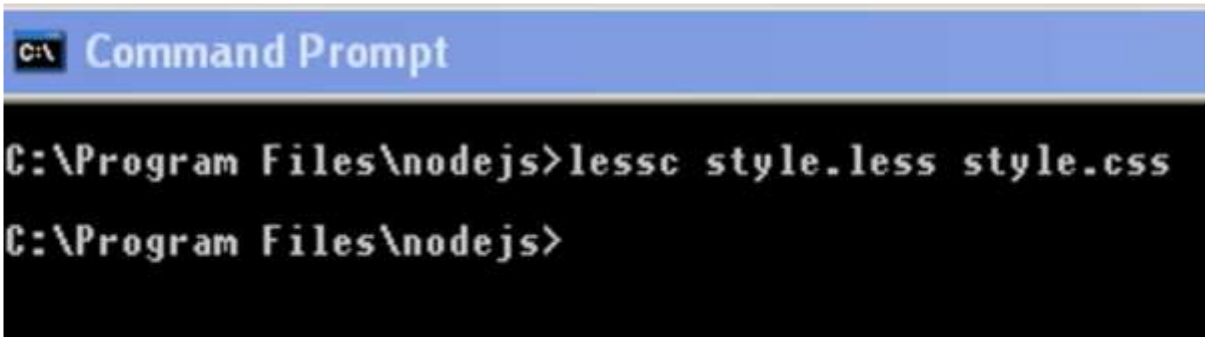
Let us now create a file *style.less* which is quite similar to CSS, the only difference is that it will be saved with *.less* extension. Both the files, *.html* and *.less* should be created inside the folder **nodejs**.

style.less

```
@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
```



```
C:\Program Files\nodejs>lessc style.less style.css

C:\Program Files\nodejs>
```

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:

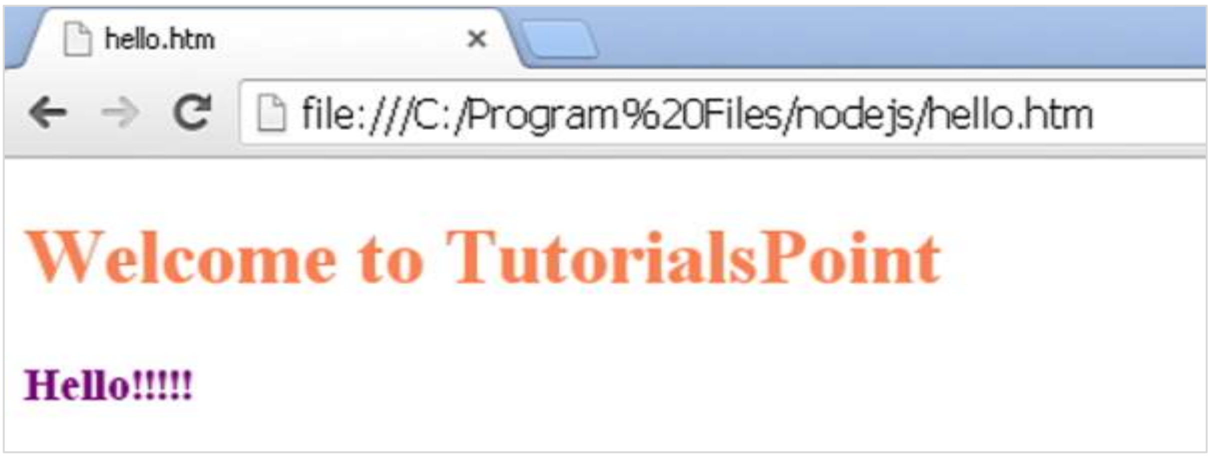
style.css

```
h1 {
  color: #FF7F50;
}
h3 {
  color: #800080;
}
```

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

Hello!!!!!

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>
<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 web site.</p>
   </div>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

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

```
 }
  p{
       font-size: 25px;
       color:#3C7949;
  }

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

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:

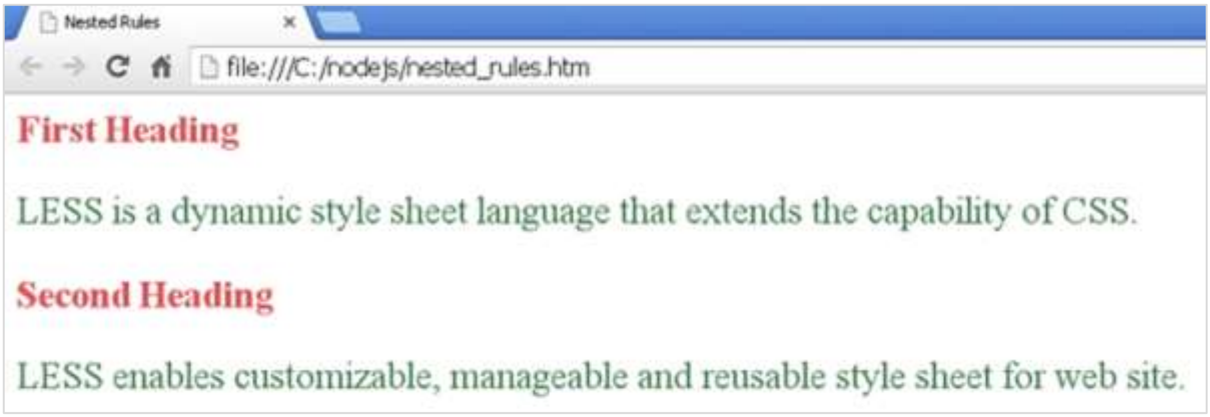
style.css

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

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.



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>
<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

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

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:

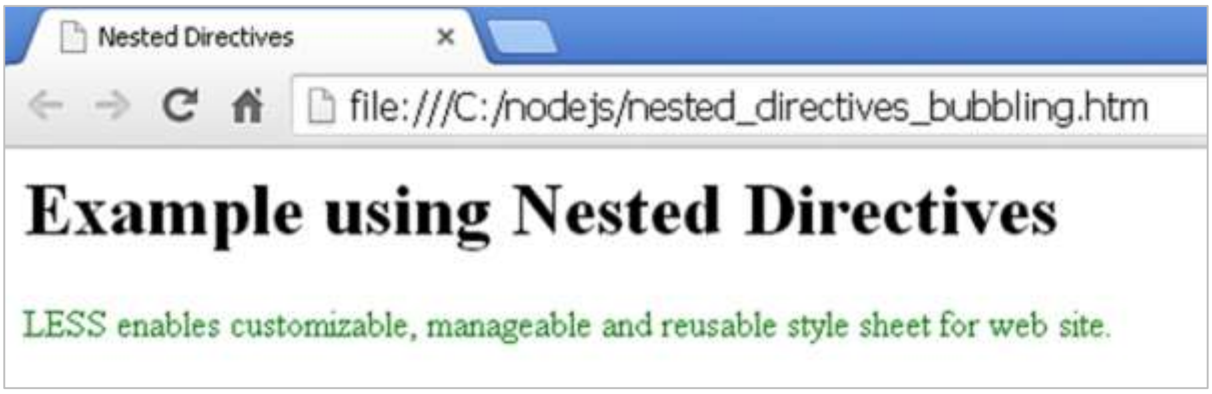
style.css

```
@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.

5. LESS — Operations

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>
<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

```
@fontSize: 10px;
.myclass {
 font-size: @fontSize * 2;
 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:

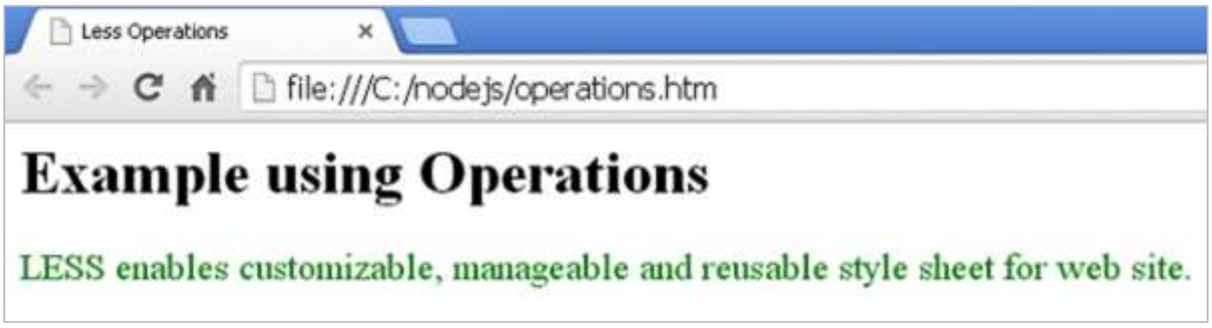
style.css

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

Output

Follow these steps to see how the above code works:

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



Example using Operations

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

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>
<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

```
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

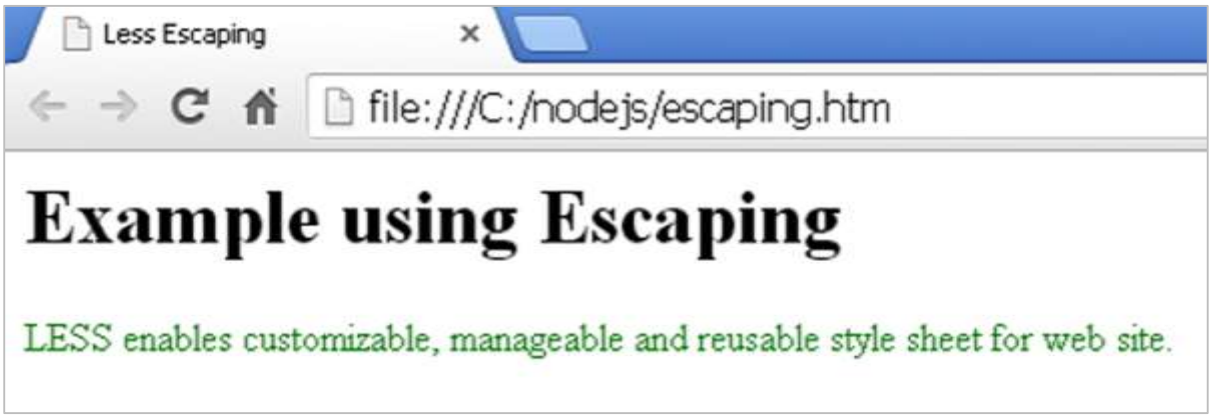
```
p {
  color: green;
}
```

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

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

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>
<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

```
@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:

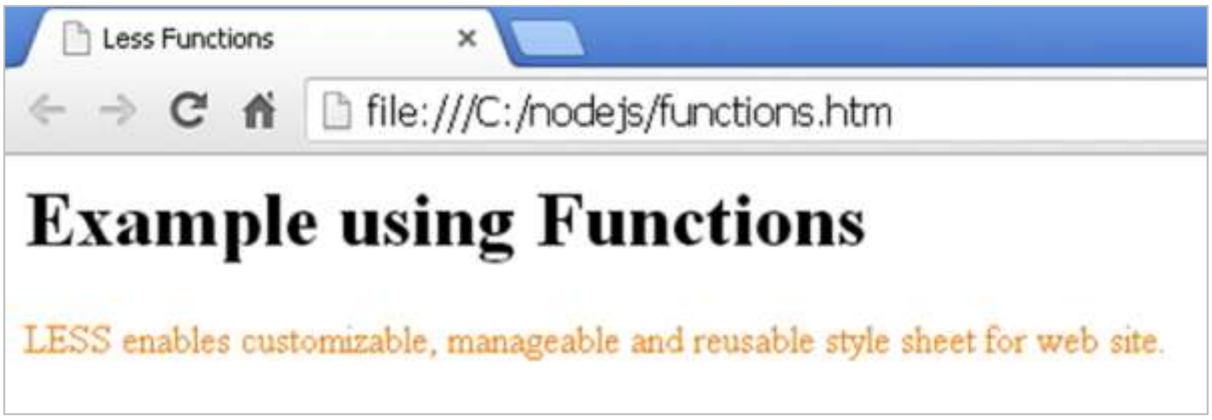
style.css

```
.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.

8. LESS — Namespaces and Accessors

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>
<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

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

.myclass {
  .class1 > .class2 > .val(20px);
}
```

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:

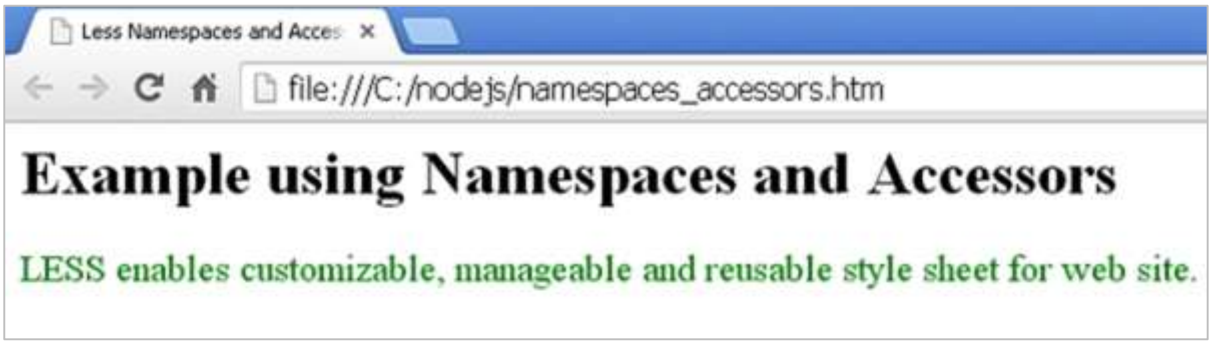
style.css

```
.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.


Less Namespaces and Acces
file:///C:/nodejs/namespaces_accessors.htm

Example using Namespaces and Accessors

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

9. LESS — Scope

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>
<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

```
@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:

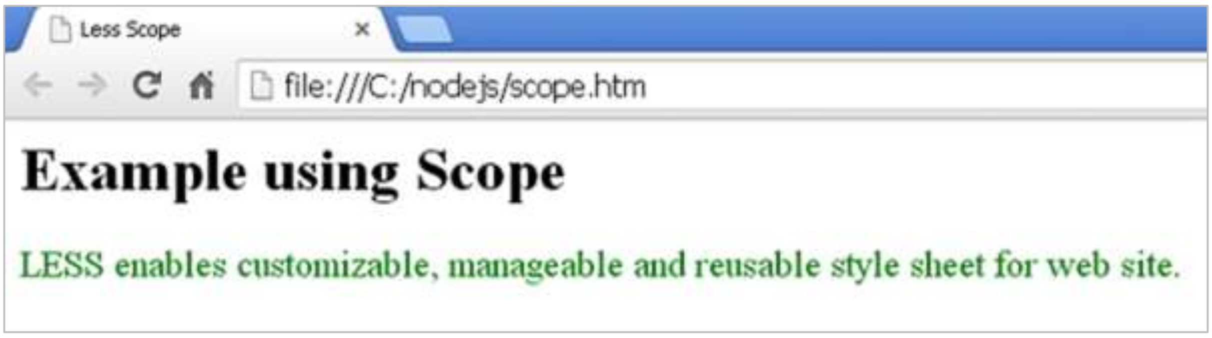
style.css

```
.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

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

10. LESS — Comments

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>
<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

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

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:

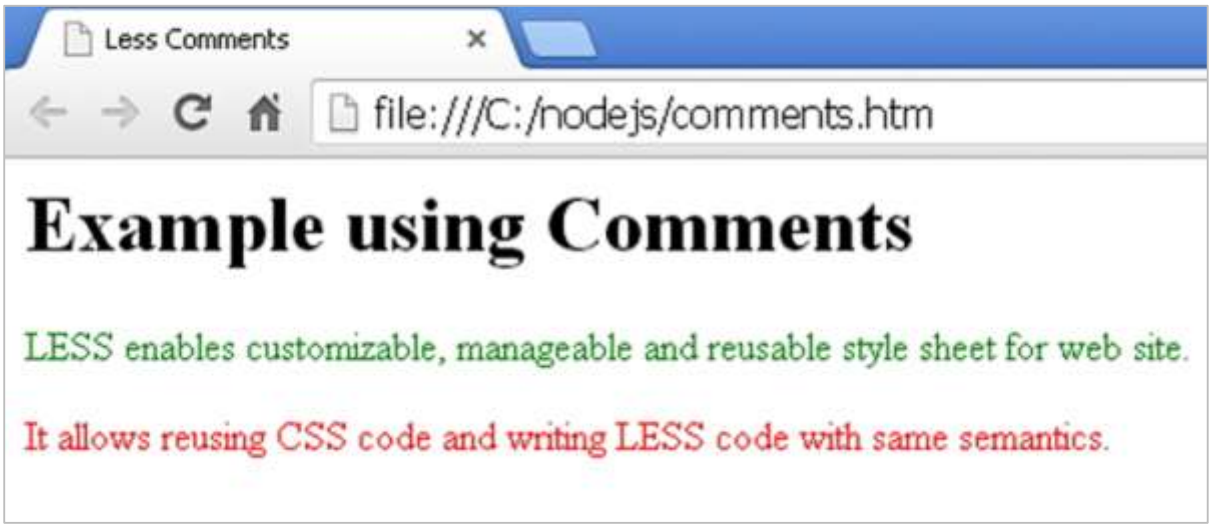
style.css

```
/* It displays the
green color! */
.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>
<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

```
.myclass{
    color: #FF8000;
}
.myclass1{
    color: #5882FA;
}
```

Now create the *style.less* file.

style.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

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:

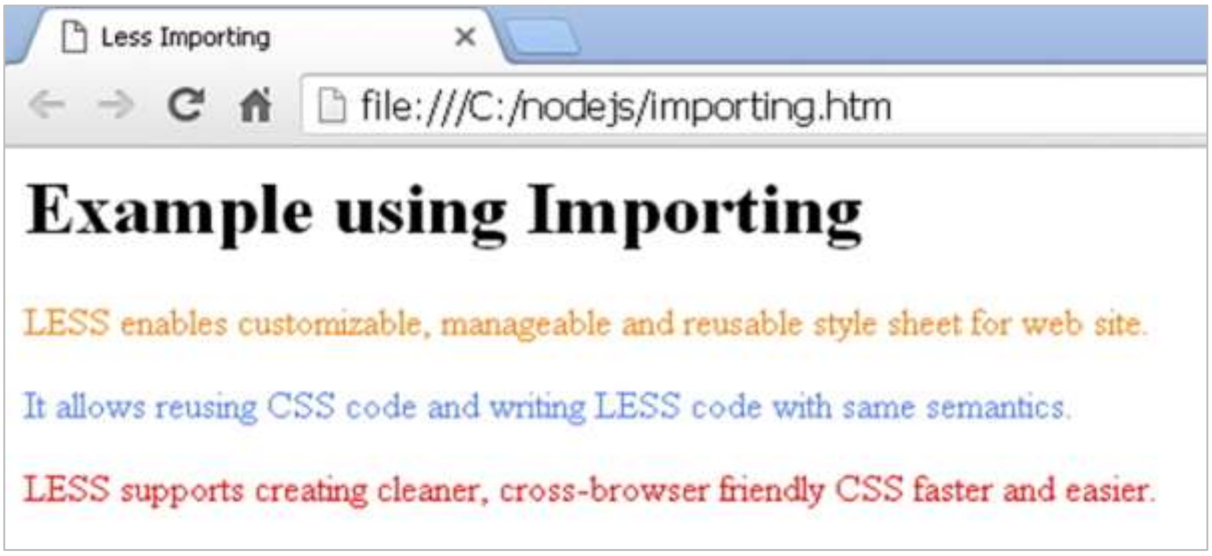
style.css

```
.myclass {
  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

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.

12. LESS — Variables

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.

S.NO.	Variables usage & Description
1	**Overview** Repetition of same value many times can be avoided by the use of *variables*.
2	**Variable Interpolation** The variables can also be used in other places like *selector names*, *property names*, *URLs* and *@import* statements.
3	**Variable Names** We can define variables with a variable name consisting of a value.
4	**Lazy Loading** In lazy loading, variables can be used even when they are not.
5	**Default Variables** 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.

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>
<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

```
@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:

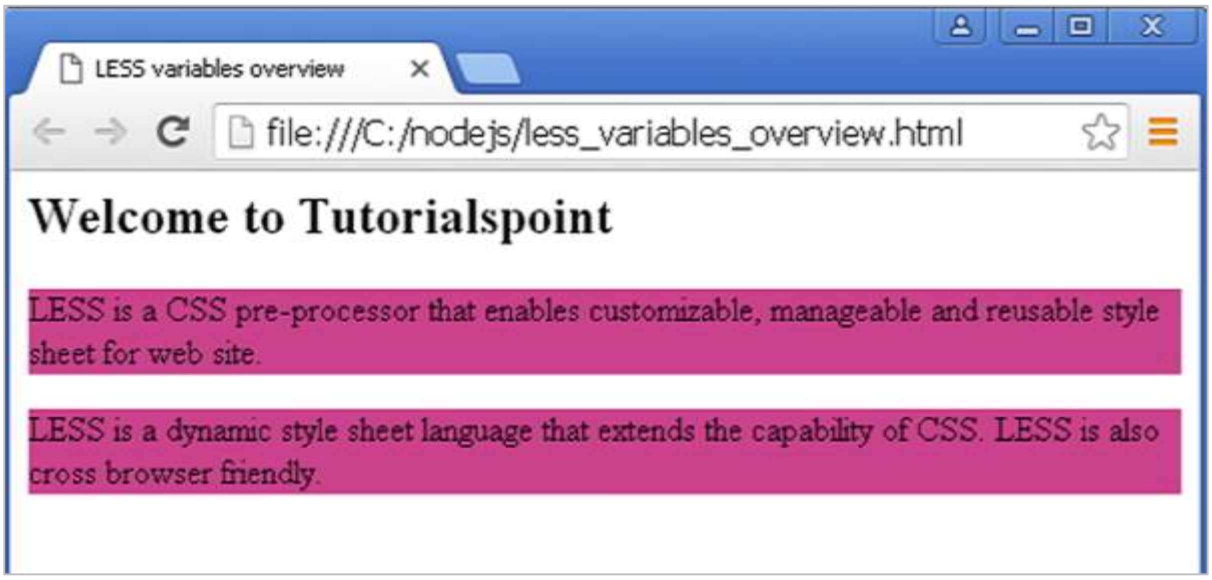
style.css

```
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.



Welcome to Tutorialspoint

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

LESS is a dynamic style sheet language that extends the capability of CSS. LESS is also cross browser friendly.

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.

S.NO.	Variables usage & Description
1	**Selectors** The selector can reference any variable and it is built during the compile time.
2	**URLs** The variables can be used to hold URLs.
3	**Import Statements** An import statement can have a variable which holds a path.
4	**Properties** The variables can be referenced by properties.

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>
   <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>

      <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

```
@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:

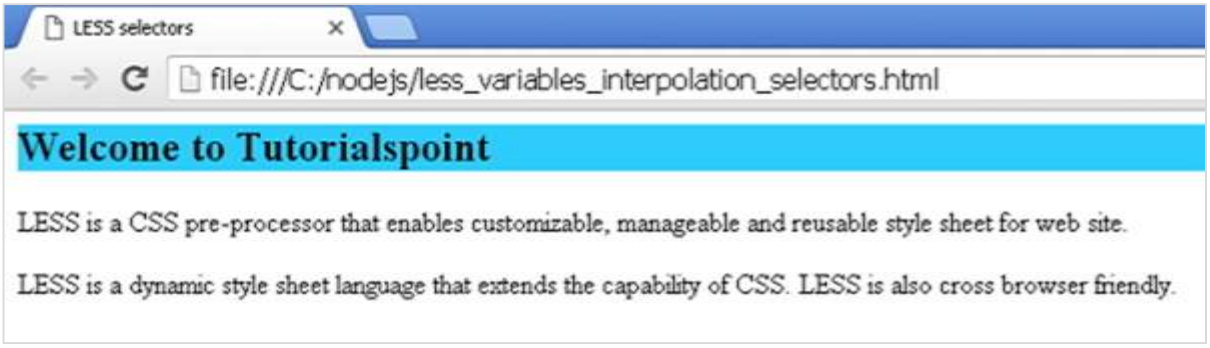
style.css

```
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 selectors
file:///C:/nodejs/less_variables_interpolation_selectors.html

Welcome to Tutorialspoint

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

LESS is a dynamic style sheet language that extends the capability of CSS. LESS is also cross browser friendly.

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>
<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

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

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

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:

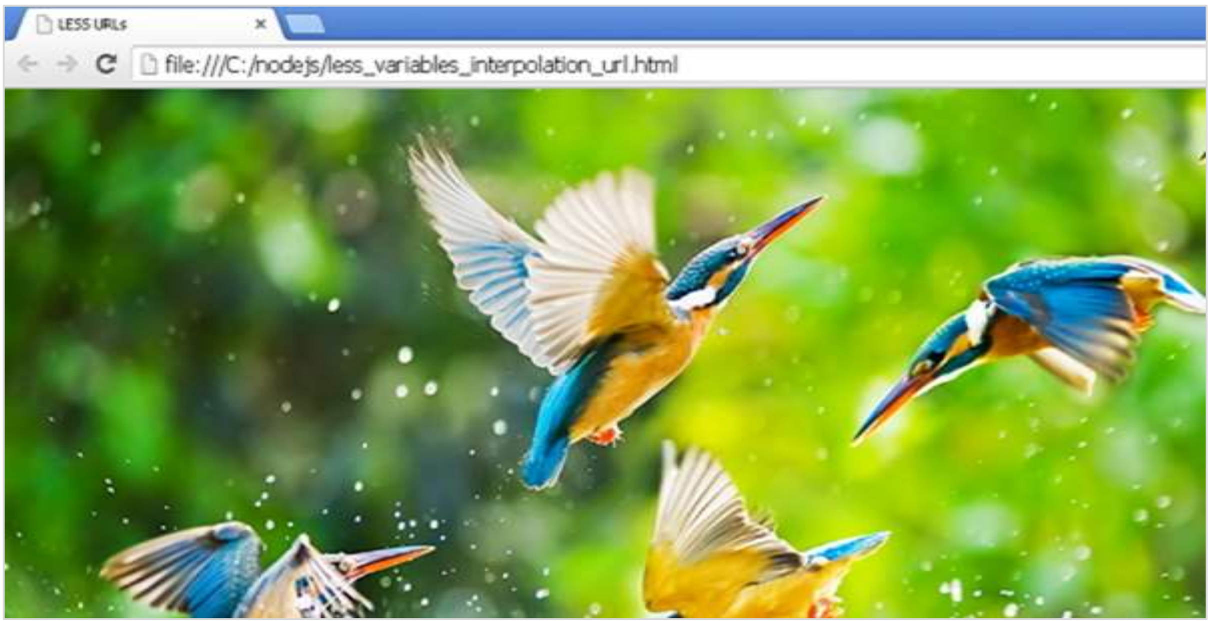
style.css

```
.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>
<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

```
@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

```
.myclass{
background: #C0C0C0;
}
```

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:

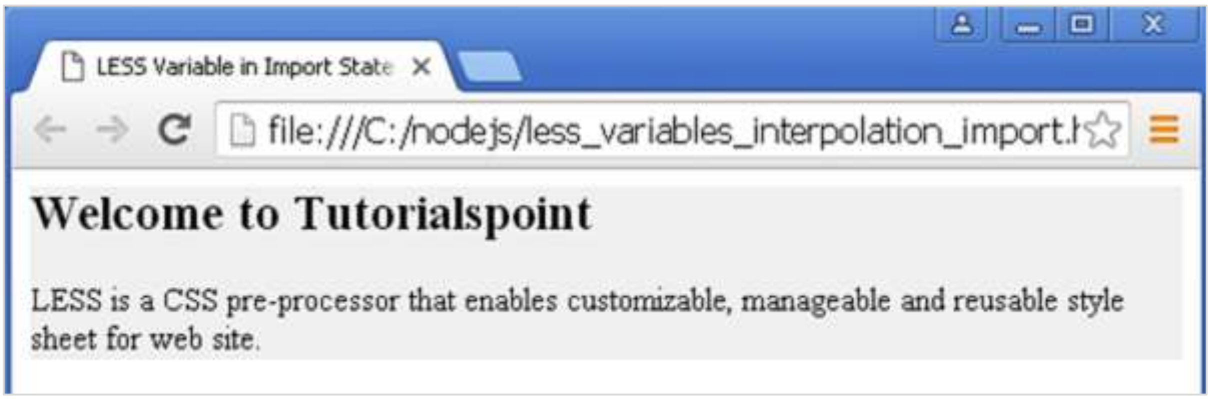
style.css

```
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 Variable in Import State
file:///C:/nodejs/less_variables_interpolation_import.h

Welcome to Tutorialspoint

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

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>
<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*.

style.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:

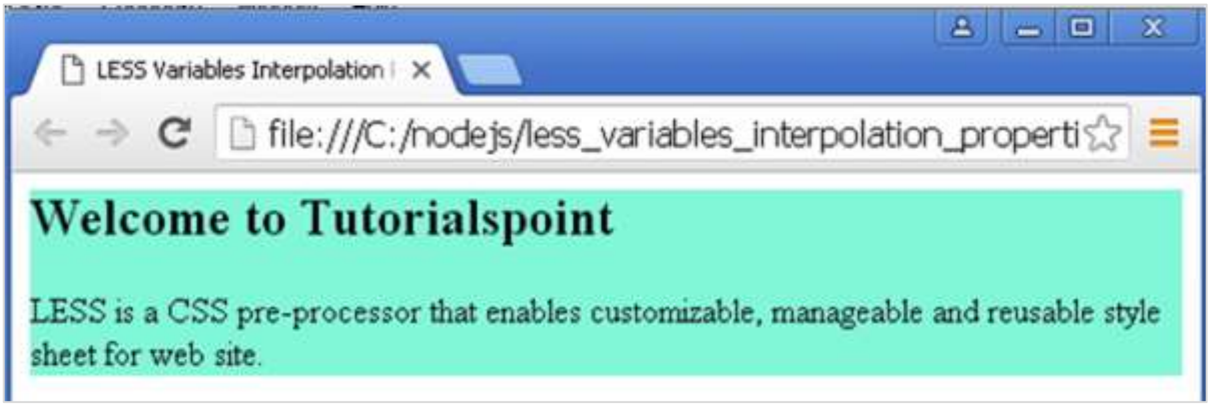
style.css

```
.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.


LESS Variables Interpolation
file:///C:/nodejs/less_variables_interpolation_properti

Welcome to Tutorialspoint

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

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>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>LESS Variable Names</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*.

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:

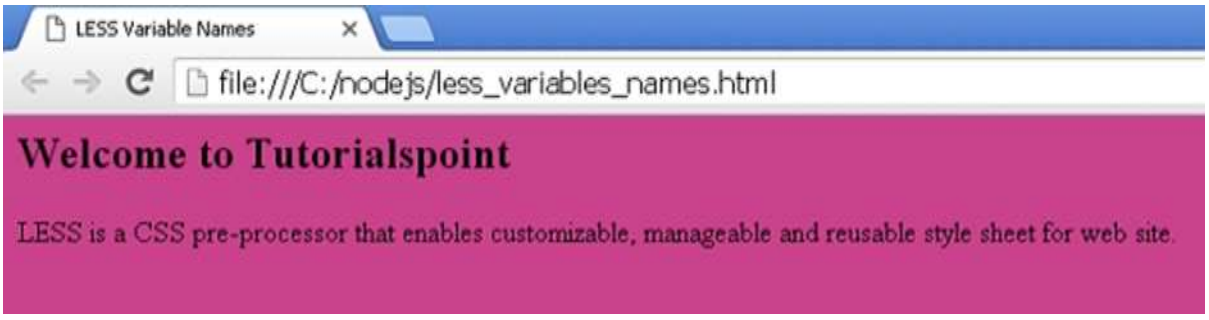
style.css

```
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>
<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

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

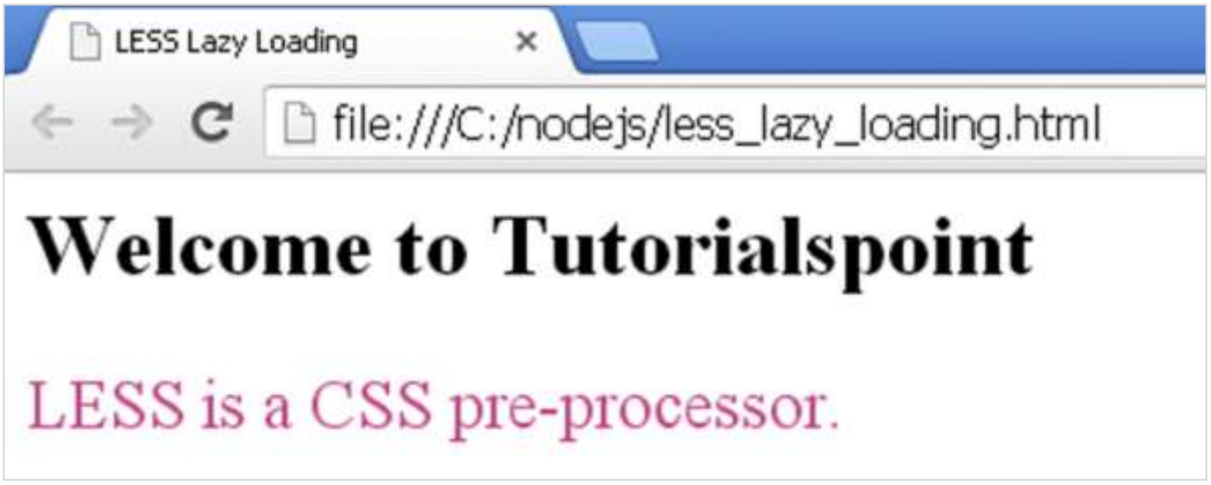
style.css

```
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.



Welcome to Tutorialspoint

LESS is a CSS pre-processor.

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>
<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

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

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:

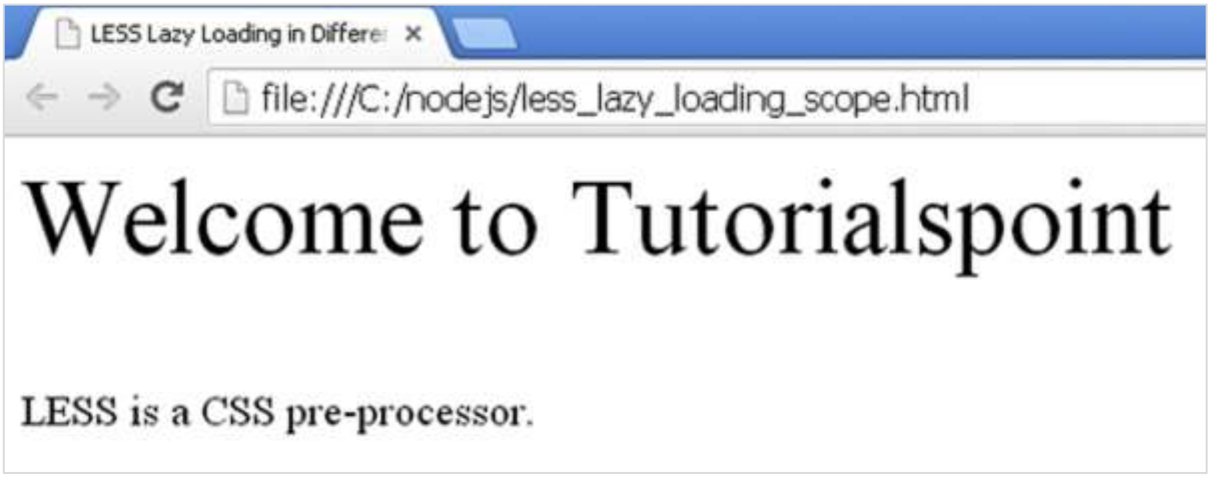
style.css

```
.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 Lazy Loading in Differe
file:///C:/nodejs/less_lazy_loading_scope.html

Welcome to Tutorialspoint

LESS is a CSS pre-processor.

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>
<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

```
@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 path:

lib.less

```
@color: blue;
```

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:

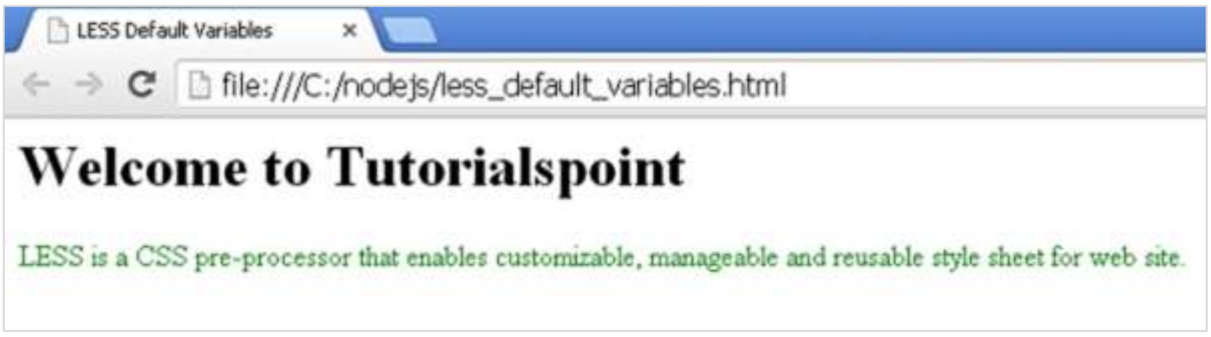
style.css

```
p {
  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.



Welcome to Tutorialspoint

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

13. LESS — Extend

Extend is a LESS pseudo class which extends other selector styles in one selector by using **:extend** selector.

Example

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

extend_syntax.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <div class="style">
     <h2>Welcome to TutorialsPoint</h2>
     <p>Hello!!!!!</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
h2 {
  &:extend(.style);
  font-style: italic;
}
.style {
  background: green;
}
```

You can compile the *extend.less* file to *extend.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:

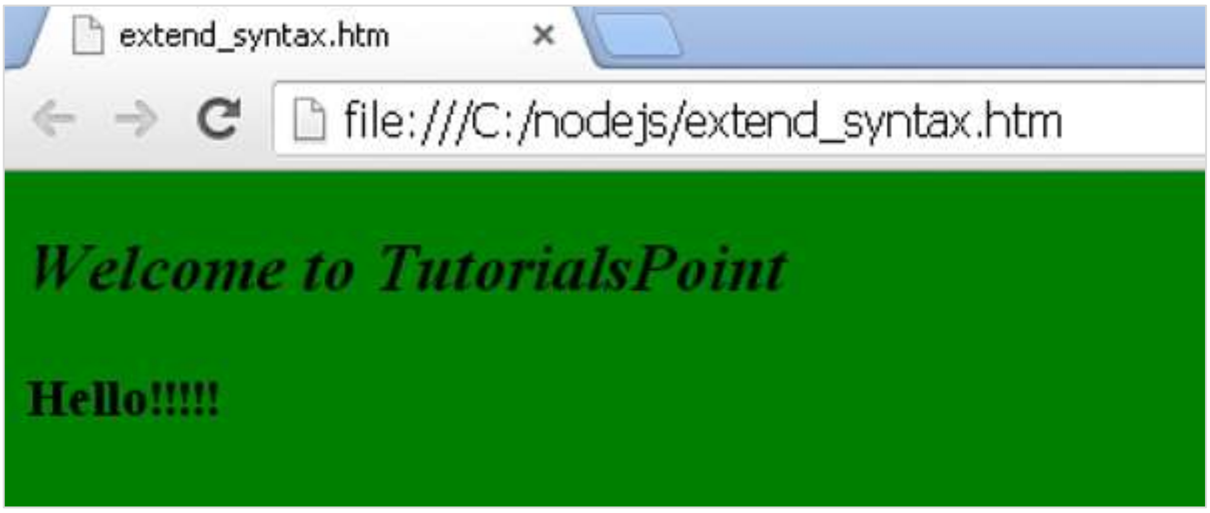
style.css

```
h2 {
  font-style: italic;
}
.style,
h2 {
  background: blue;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

Hello!!!!!

Extend Syntax

Extend is placed into ruleset or attached to a selector. It is similar to a pseudo class containing one or more classes, which are separated by comma. Using the optional keyword **all**, each selector can be followed.

Example

The following example demonstrates the use of *extend syntax* in the LESS file:

extend_syntax.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <div class="style">
    <h2>Welcome to TutorialsPoint</h2>
    <div class="container">
      <p>Hello!!!!!</p>
    </div>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style:extend(.container, .img)
{
  background: #BF70A5;
}
.container {
  font-style: italic;
}
.img{
   font-size: 30px;
 }
```

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:

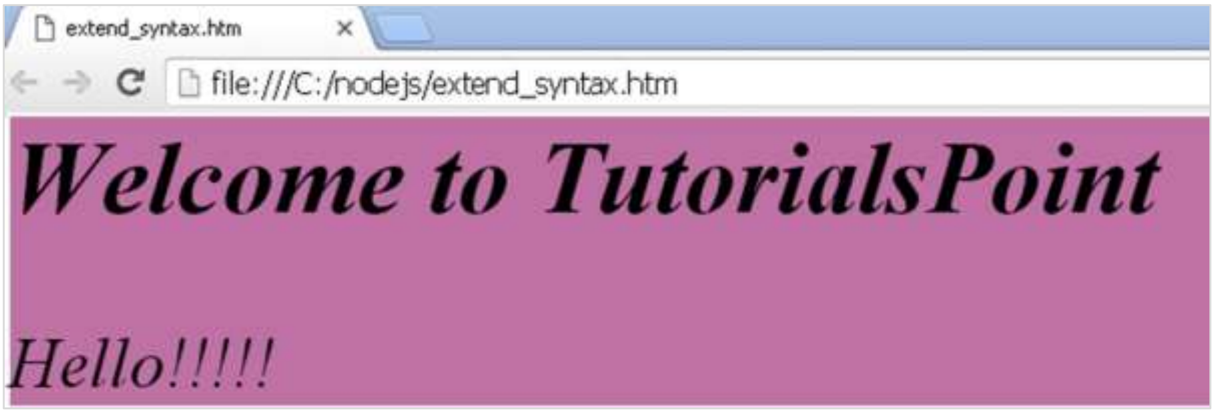
style.css

```
.style {
  background: #BF70A5;
}
.container,
.style {
  font-style: italic;
}
.img,
.style {
  font-size: 30px;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

Hello!!!!!

The following table lists all the types of extend syntax which you can use in LESS:

S.NO.	Types & Description
1	**Extend Attached to Selector** Extend is connected to a selector which looks similar to a pseudo class with selector as parameter.
2	**Extend Inside Ruleset** The **&:extend(selector)** syntax can be put inside the body of ruleset.
3	**Extending Nested Selectors**

	Nested selectors are matched using the *extend* selector.
4	**Exact Matching with Extend** By default, **extend** looks for the exact match between the selectors.
5	**nth Expression** The form of nth expression is important in extend otherwise, it treats selector as different.
6	**Extend "all"** When the keyword *all* is identified at last in the **extend** argument then LESS matches that selector as part of another selector.
7	**Selector Interpolation with Extend** The **extend** can be connected to interpolated selector.
8	**Scoping/Extend inside @media** **Extend** matches the selector only that is present inside the same media declaration.
9	**Duplication Detection** It cannot detect the duplication of selectors.

LESS – Extend Attached to the Selector

Description

Extend is connected to a selector which looks similar to a pseudo class with selector as parameter. When the ruleset has many selectors, then keyword extend can be applied on any of the selectors. The following format can be used to define the *extend* in code.

- Extend after the selector [Eg: *pre:hover:extend(div pre)*]
- Allows space between selector and extend [Eg: *pre:hover :extend(div pre)*]
- Allows multiple extends [Eg: *pre:hover:extend(div pre):extend(.bucket tr)* or *pre:hover:extend(div pre, .bucket tr)*]
- Extend must be defined at the end of the selector. *pre:hover:extend(div pre).nth-child(odd)* type is not allowed.

Example

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

extend_syntax.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <div class="style">
     <h2>Welcome to TutorialsPoint</h2>
     <div class="container">
       <p>Hello!!!!!</p>
     </div>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style,
.container:extend(.img){
   background: #BF70A5;
}
.img{
   font-size: 45px;
   font-style: italic;
}
```

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:

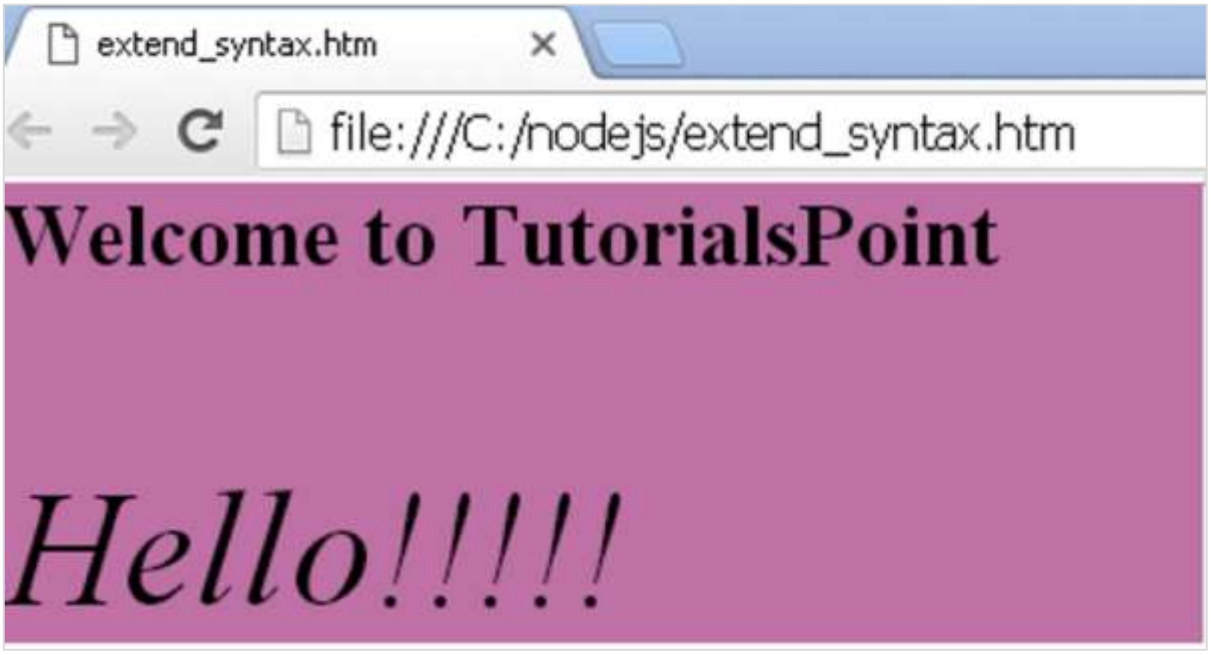
style.css

```
.style {
  background: #BF70A5;
}
.container,
.style {
  font-style: italic;
}
.img,
.style {
  font-size: 45px;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

Hello!!!!!

LESS – Extend Inside Ruleset

Description

The **&:extend(selector)** syntax can be put inside the body of ruleset. It is shortcut of placing extend into every single selector of the ruleset.

Example

The following example demonstrates the use of extend inside ruleset in the LESS file:

extend_syntax.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <div class="style">
      <h1>Welcome to TutorialsPoint</h1>
      <div class="container">
         <h2>Hello!!!!!</h2>
      </div>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.container,
.style {
     &:extend(.img);
}
.img{
    font-style: italic;
    background-color: #7B68EE;
 }
```

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:

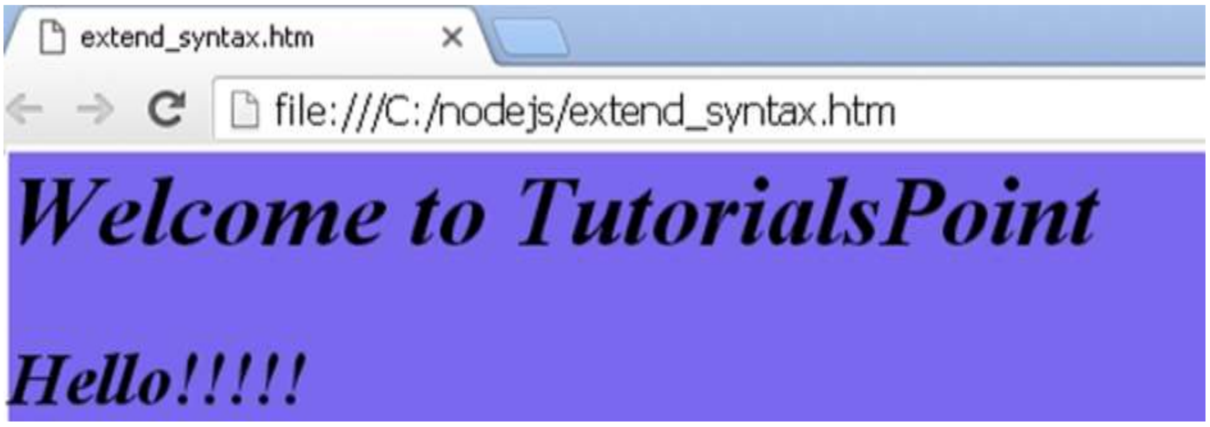
style.css

```
.img,
.container,
.style {
  font-style: italic;
  background-color: #7B68EE;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

Hello!!!!!

LESS — Extending Nested Selectors

Description

Nested selectors are matched using the *extend* selector.

Example

The following example demonstrates use of extending nested selectors in the LESS file:

```
extend_syntax.htm
<!doctype html>
<head>
     <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="style">
     <h3>Hello!!!!!</h3>
</div>
<p class="img">Welcome to TutorialsPoint</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style {
  h3{
     color: #BF70A5;
     font-size: 30px;
}
}
.img:extend(.style h3){}
```

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:

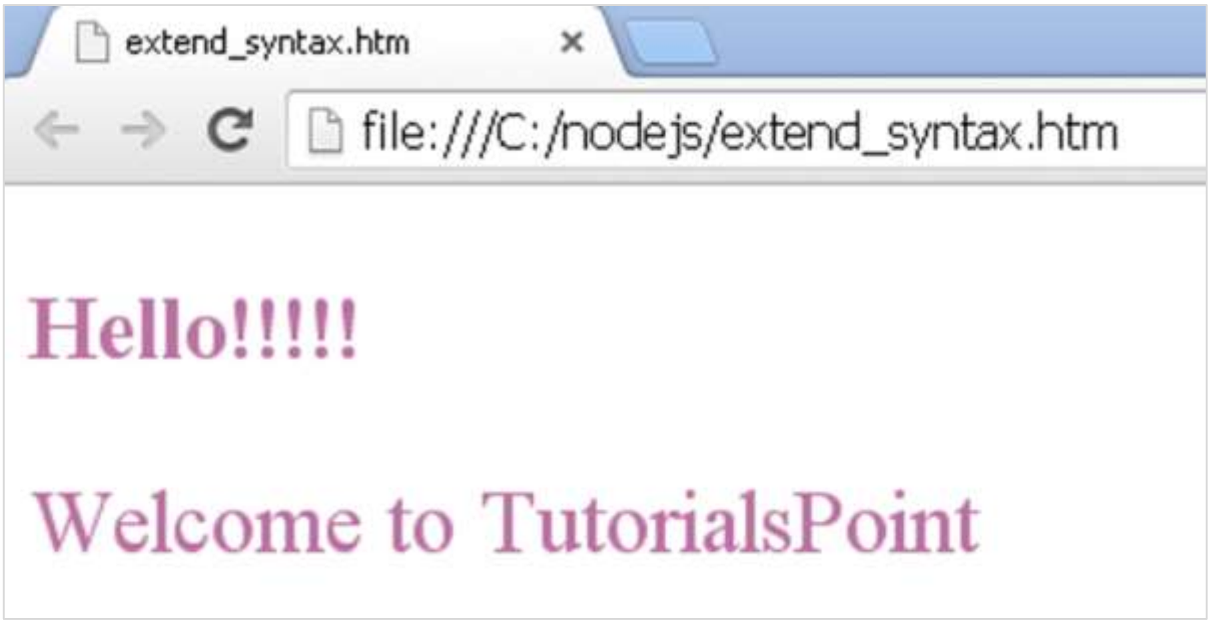
style.css

```
.style h3,
.img {
     color: #BF70A5;
     font-size: 30px;
}
```

Output

Follow these steps to see how the above code works:

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



Hello!!!!!

Welcome to TutorialsPoint

LESS — Exact Matching with Extend

Description

By default, the **extend** look for the exact match between the selectors. The extend does not matter when it comes to two nth - expressions having same meaning, but it only looks for the same order form as defined for the selector to match.

Example

The following example demonstrates the use of exact matching with **extend** in the LESS file:

extend_syntax.htm

```
<!doctype html>
<head>
     <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>

<div class="style">
```

```
     <h3>Hello!!!!!</h3>
</div>
<h4 class="img">Welcome to TutorialsPoint</h4>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style h3,
h3.style{
     color: #BF70A5;
     font-style: italic;
}
.img:extend(.style h3){}
```

The *.style h3* should be defined in the same way in extend as defined for selector. If you define in extend as *.style* then extend treats this as different.

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:

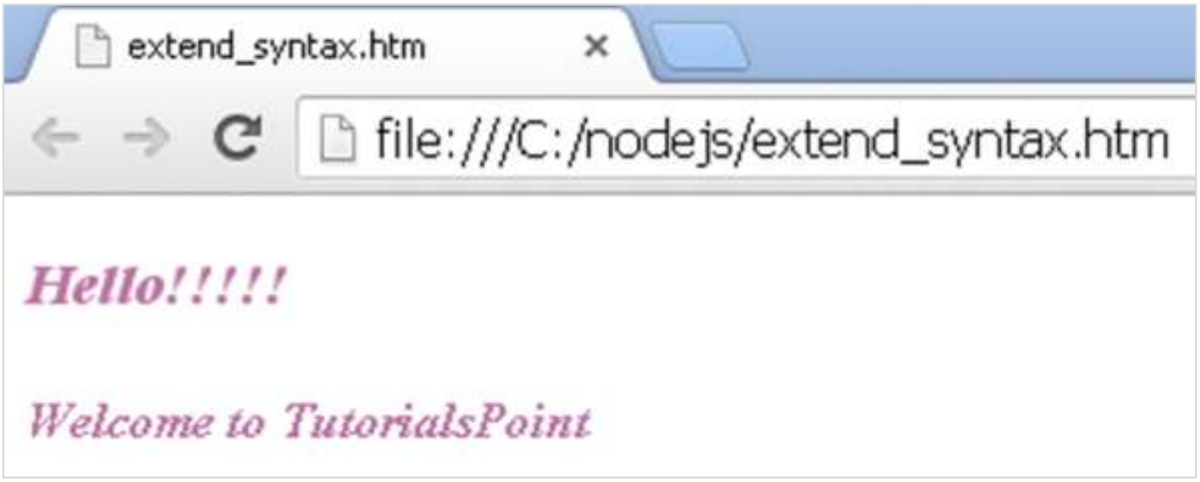
style.css

```
.style h3,
h3.style,
.img {
  color: #BF70A5;
  font-style: italic;
}
```

Output

Follow these steps to see how the above code works:

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



Hello!!!!!

Welcome to TutorialsPoint

LESS — nth Expression

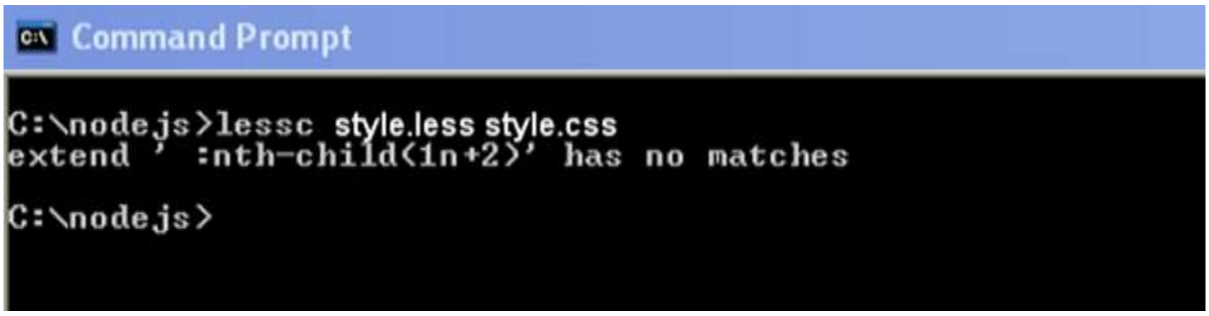
Description

The form of nth expression is important in extend, otherwise it treats the selector as different. The nth expression **1n+2** and **n+2** are equivalent but extend treats this expression as different.

For instance, create one LESS file with the following code:

```
:nth-child(n+2)
{
color: #BF70A5;
  font-style: italic;
}
.child:extend(:nth-child(1n+2)){}
```

When we compile the above code in command prompt, then you will get an error message as shown below.



```
C:\nodejs>lessc style.less style.css
extend ' :nth-child(1n+2)' has no matches

C:\nodejs>
```

After compiling, you will get the following CSS code.

```
:nth-child(n+2) {
  color: #BF70A5;
  font-style: italic;
}
```

In attribute selector, the quote type is not important as you can see it in the following example:

Example

The following example demonstrates the use of the **n**th expression in the LESS file:

extend_syntax.htm

```
<!doctype html>
<head>
     <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="style">
   <h2>Hello!!!!!</h2>
</div>
<p class="img">Welcome to TutorialsPoint</p>
</body>
</html>
Next, create file style.less
style.less
[title=tutorialspoint] {
  font-style: italic;
}
[title='tutorialspoint'] {
 font-style: italic;
}
[title="tutorialspoint"] {
  font-style: italic;
}
.style:extend([title=tutorialspoint]) {}
.container:extend([title='tutorialspoint']) {}
.img:extend([title="tutorialspoint"]) {}
```

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:

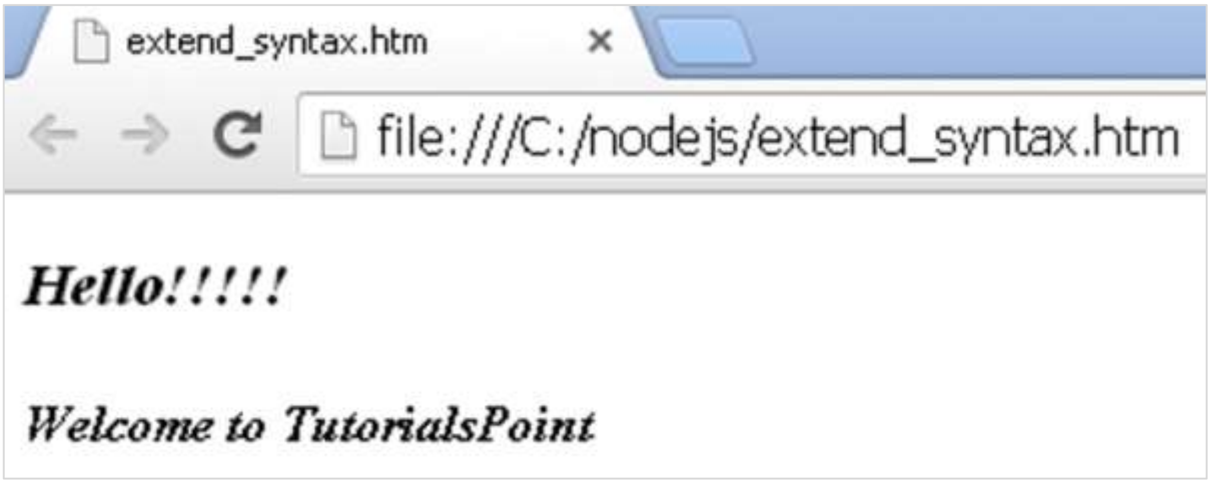
style.css

```
[title=tutorialspoint],
.style,
.container,
.img {
  font-style: italic;
}
[title='tutorialspoint'],
.style,
.container,
.img {
  font-style: italic;
}
[title="tutorialspoint"],
.style,
.container,
.img {
  font-style: italic;
}
```

Output

Follow these steps to see how the above code works:

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



Hello!!!!!

Welcome to TutorialsPoint

LESS – Extend "all"

Description

When the keyword *all* is identified at last in the **extend** argument, then LESS matches that selector as part of another selector. The selector part which is matched will be replaced with extend, making a new selector.

Example

The following example demonstrates the use of all keyword in the LESS file:

extend_all.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <div class="container">
      <h3>Welcome to TutorialsPoint</h3>
      <div class="style">
      <ul class="nav">
         <li>SASS </li>
```

```
          <li>LESS</li>
       </ul>
       </div>
    </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style.nav,
.nav h3 {
  color: orange;
}
.nav {
  &:hover {
    color: green;
  }
}
.container:extend(.nav all) {}
```

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:

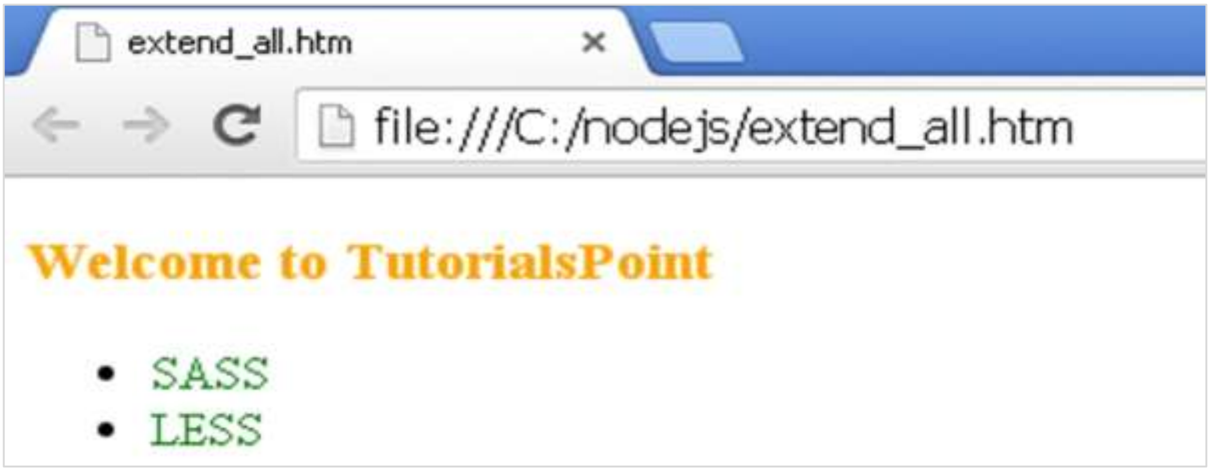
style.css

```
.style.nav,
.nav h3,
.style.container,
.container h3 {
  color: orange;
}
.nav:hover,
.container:hover {
   color: green;
}
```

Output

Follow these steps to see how the above code works:

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



extend_all.htm

file:///C:/nodejs/extend_all.htm

Welcome to TutorialsPoint

- SASS
- LESS

LESS — Selector Interpolation with Extend

Description

The **@{variable}** symbol is used as part of the variable name, id and class name. **Extend** does not have the ability to match selector with variables. Extend can be connected to the interpolated selector.

Example

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

extend_syntax.htm

```
<!doctype html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2 class="style">This is a simple example of Selector Interpolation</h2>
   <p class="style">This is a simple example of Selector Interpolation</p>
```

```
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style {
  color: #BF70A5;
  font-style: italic;
}
@{variable}:extend(.style) {}
@variable: .selector;
```

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:

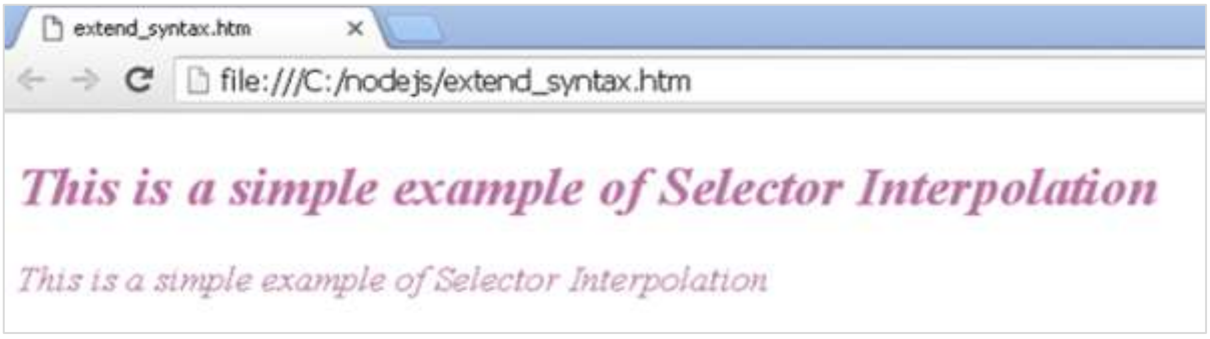
style.css

```
.style,
.selector {
  color: #BF70A5;
  font-style: italic;
}
```

Output

Follow these steps to see how the above code works:

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



This is a simple example of Selector Interpolation

This is a simple example of Selector Interpolation

LESS — Scoping/Extend inside @media

Description

Inside the media declaration, **extend** should be written. The Extend matches the selector only that are present inside the same media declaration. The Extend present in media declaration does not match with the selector present inside the nested declaration.

Example

The following example demonstrates the use of scoping extend inside media in the LESS file:

extend_syntax.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
  <h2>Example using extend inside media</h2>
  <img src="/less/images/less-extend/nature.jpg" class="style">
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
@media screen {
  .style {
    width:500px;
  }
  @media (min-width: 1023px) {
    .style {
      width:500px;
    }
  }
}
.cont:extend(.style) {}
```

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:

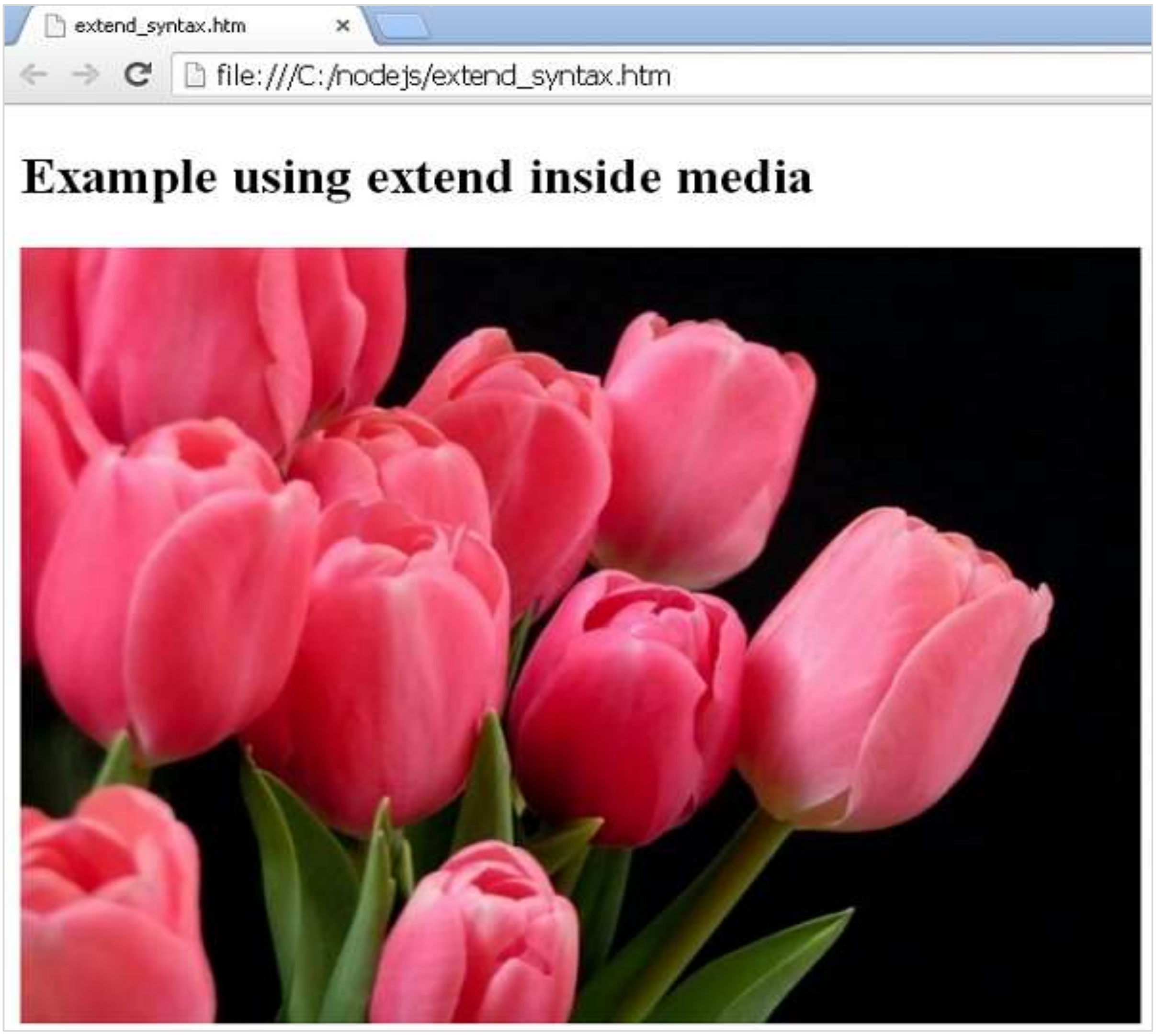
style.css

```
@media screen {
  .style,
  .cont {
    width: 500px;
  }
}
@media screen and (min-width: 1023px) {
  .style,
  .cont {
    width: 500px;
  }
}
```

Output

Follow these steps to see how the above code works:

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



Example using extend inside media

LESS – Duplication Detection

Description

It cannot detect the duplication of selectors.

Example

The following example demonstrates the use of duplication detection in the LESS file:

extend_syntax.htm

```
<!doctype html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="cont-main">
   <p>Welcome to TutorialsPoint</p>
   <div class="style">
     <ul>
       <li>SASS </li>
       <li>LESS</li>
     </ul>
   </div>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.cont-main,
.style {
  font-family: "Comic Sans MS";
  font-size: 30px;
}
.cont-main{
  font-size: 30px;
}
.cont:extend(.cont-main, .style) {}
```

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:

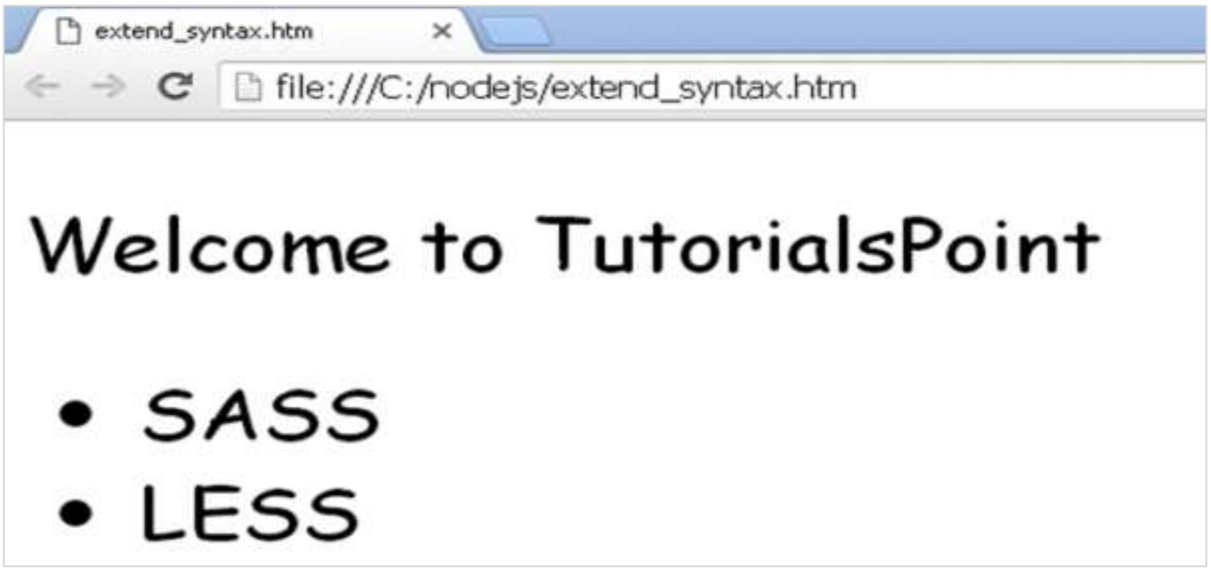
style.css

```
.cont-main,
.style,
.cont,
.cont {
  font-family: "Comic Sans MS";
  font-size: 30px;
}
.cont-main,
.cont {
  font-size: 30px;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

- SASS
- LESS

Following are the types of Use Cases for Extend

S.NO.	Types & Description
1	**Classic Use Case** Classic use case is used to avoid adding the base class in LESS.
2	**Reducing CSS Size** Extend is used to move the selector as far as the properties you want to use; this helps in reducing the css generated code.
3	**Combining Styles/ A More Advanced Mixin** Using extend we can combine the same styles of a particular selectors into other selector.

LESS — Classic Use Case

Description

Classic use case is used to keep away adding the base class in LESS. The values written in base class are avoided when the same values are overridden inside the other class.

Example

The below example describes the use of classic use case in the LESS file:

extend_syntax.htm

```
<!doctype html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="cont">
  <h2>Welcome to TutorialsPoint</h2>
  <p>The largest Tutorials Library on the web.</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style {
  font-family: "Times New Roman";
  font-style:italic;
}
.cont {
  &:extend(.style);
font-family: "Comic Sans MS";
}
```

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:

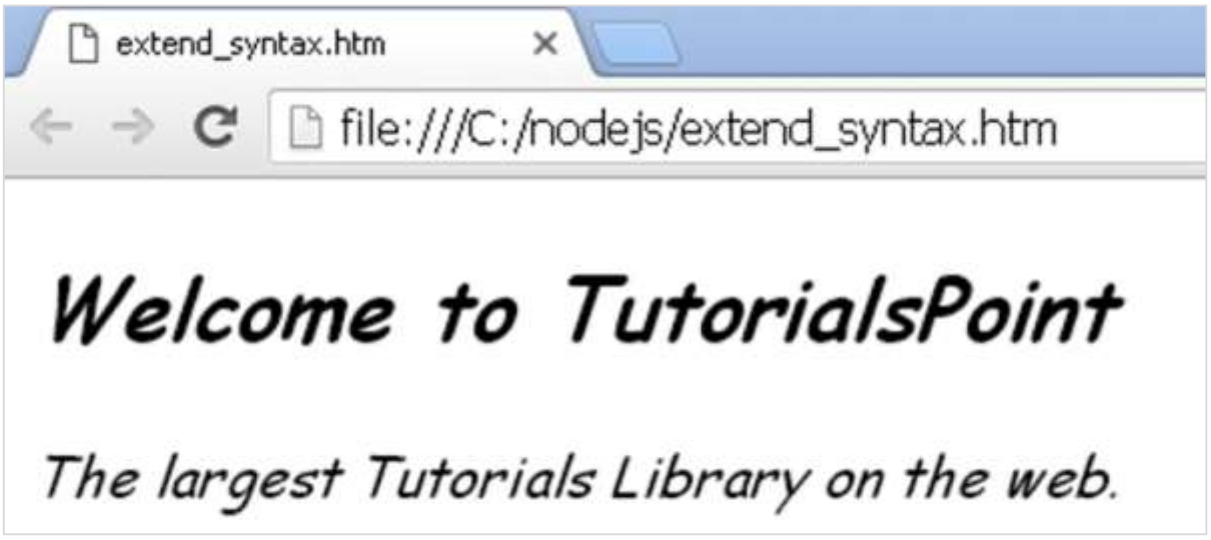
style.css

```
.style,
.cont {
  font-family: "Times New Roman";
  font-style: italic;
}
.cont {
  font-family: "Comic Sans MS";
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

LESS – Reducing CSS Size

Description

Extend is used to move the selector as far as the properties you want to use, which helps in reducing the **css** generated code.

Example

The following example shows how to reduce the css size in a LESS file:

extend_syntax.htm

```
<!doctype html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="cont">
  <div class="style">
    <h2 class="cont">The largest Tutorials Library on the web.</h2>
    <ul>
      <li>HTML</li>
      <li>SASS</li>
      <li>LESS</li>
    </ul>
  </div>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style {
  display: inline-block;
   background-color: black;
  color: white;
}
.cont {
  &:extend(.style);
}
.nav {
  &:extend(.style);
}
```

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:

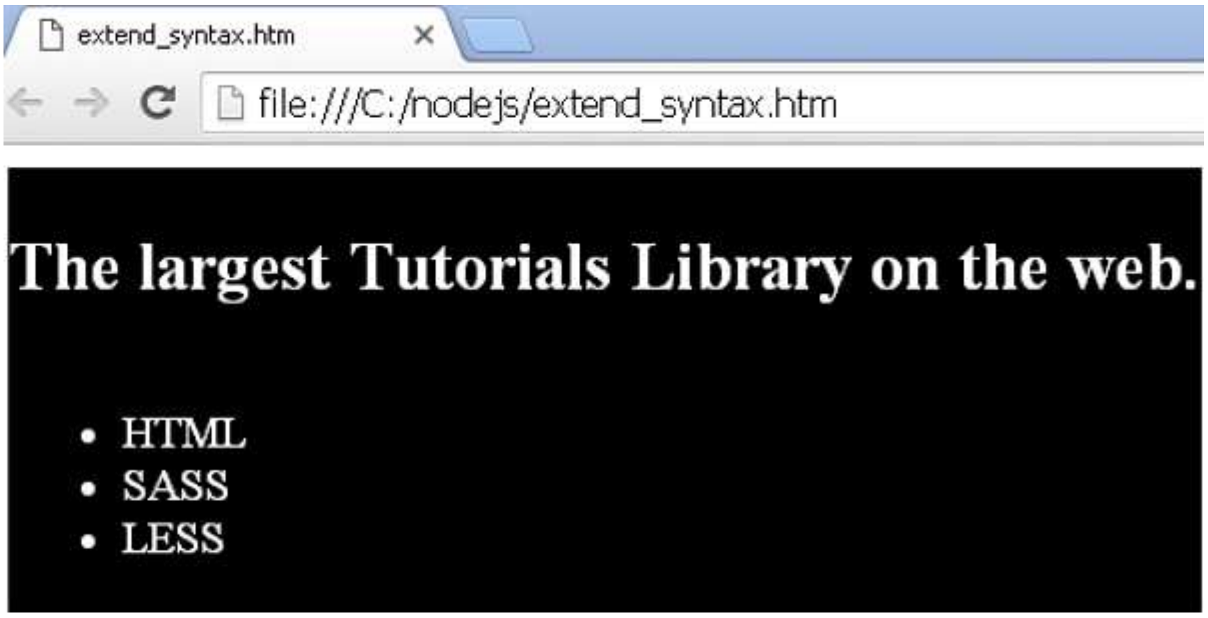
style.css

```
.style,
.cont,
.nav {
  display: inline-block;
  background-color: black;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



```
The largest Tutorials Library on the web.

  • HTML
  • SASS
  • LESS
```

LESS — Combining Styles/ A More Advanced Mixin

Description

Using **Extend**, we can combine the same styles of a particular selector into other selector.

Example

The following example describes the use of combining styles mixin in the LESS file:

extend_syntax.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="cont">
  <h2>Welcome to TutorialsPoint</h2>
  <p>The largest Tutorials Library on the web.</p>
   <ul class="nav">
     <li>HTML</li>
```

```
      <li>SASS</li>
      <li>LESS</li>
    </ul>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
ul.nav > li {
   background-color: #6DE6E6;
  color: black;
}
.cont {
  &:extend(ul.nav > li);
}
```

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:

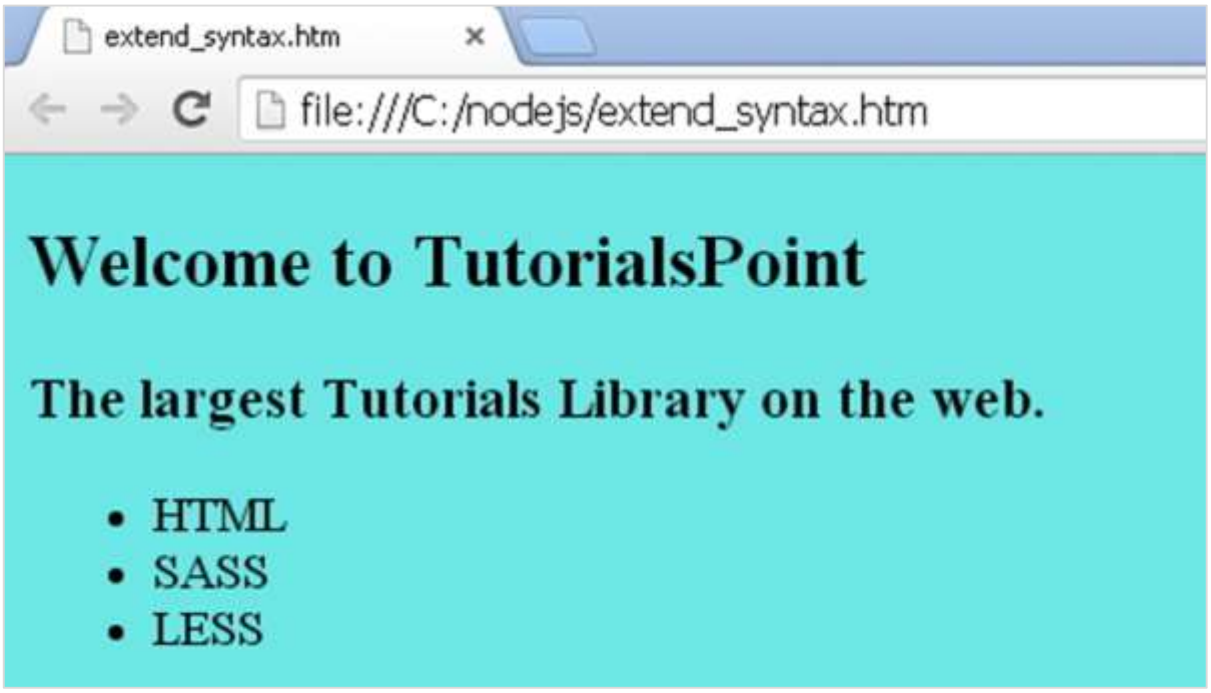
style.css

```
ul.nav > li,
.cont {
  background-color: #6DE6E6;
  color: black;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

- HTML
- SASS
- LESS

14. LESS — Mixins

Mixins are similar to functions in programming languages. Mixins are a group of CSS properties that allow you to use properties of one class for another class and includes class name as its properties. In LESS, you can declare a 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.

The following table demonstrates the use of LESS *mixins* in detail.

S.NO.	Mixins usage & Description
1	**Not Outputting the Mixin** Mixins can be made to disappear in the output by simply placing the parentheses after it.
2	**Selectors in Mixins** The mixins can contain not just properties, but they can contain selectors too.
3	**Namespaces** Namespaces are used to group the mixins under a common name.
4	**Guarded Namespaces** When guard is applied to namespace, mixins defined by it are used only when the guard condition returns true.
5	**The !important keyword** The *!important* keyword is used to override the particular property.

Example

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

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>LESS Mixins</title>
</head>
<body>
<h1>Welcome to Tutorialspoint</h1>
```

```
<p class="p1">LESS is a CSS pre-processor that enables customizable, manageable
and reusable style sheet for web site.</p>
<p class="p2">LESS is a dynamic style sheet language that extends the capability
of CSS. </p>
<p class="p3">LESS is cross browser friendly.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.p1{
  color:red;
}
.p2{
  background : #64d9c0;
  .p1();
}
.p3{
   background : #DAA520;
  .p1;
}
```

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

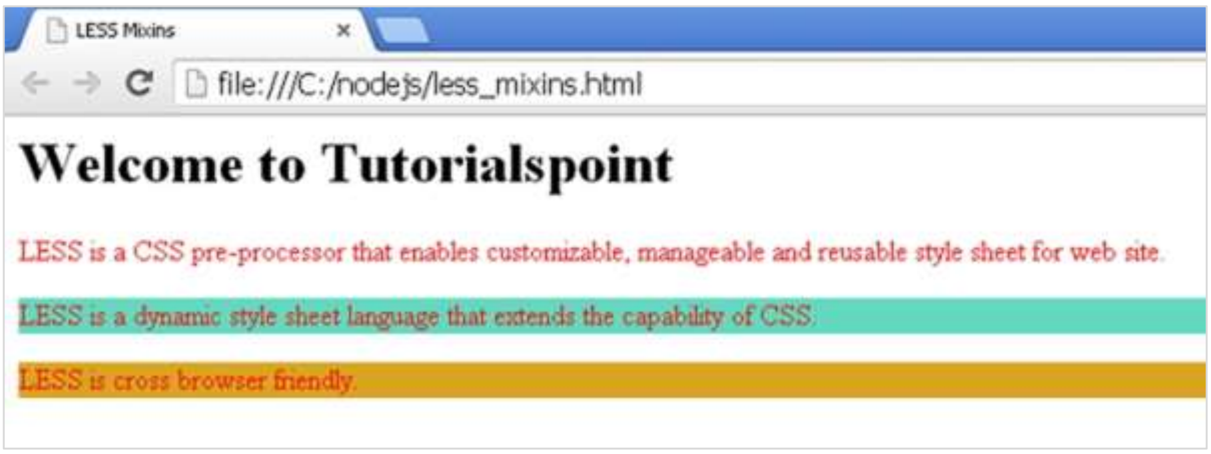
```
.p1 {
  color: red;
}
.p2 {
  background: #64d9c0;
  color: red;
}
```

```
.p3 {
  background: #DAA520;
  color: red;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **less_mixins.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 is a dynamic style sheet language that extends the capability of CSS.

LESS is cross browser friendly.

The parentheses are optional when calling mixins. In the above example, both statements **.p1();** and **.p1;** do the same thing.

LESS — Not Outputting the Mixin

Description

It is possible to create a mixin and it can be made to disappear in the output by simply placing the parentheses after it.

Example

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

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

```
<div class="myclass">
<h1>Welcome to Tutorialspoint</h1>
<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

```
.a(){
  padding-left: 100px;
}
.myclass{
  background : #64d9c0;
  .a;
}
```

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:

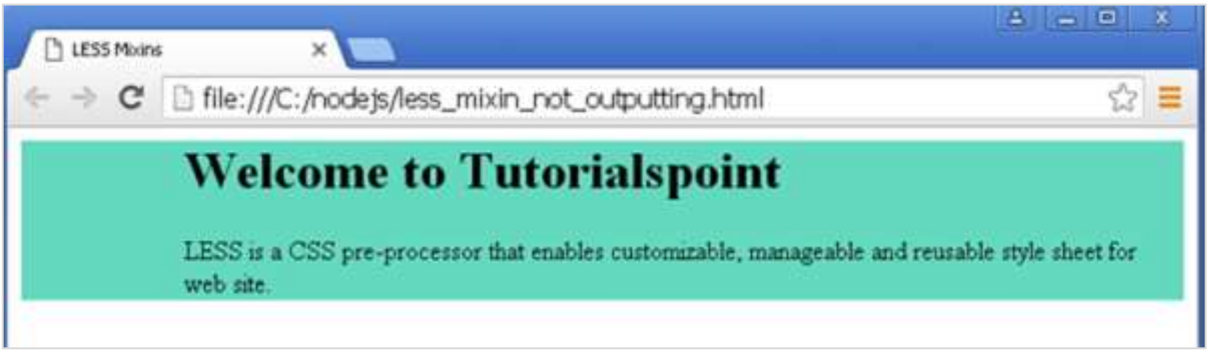
style.css

```
.myclass {
  background: #64d9c0;
  padding-left: 100px;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **less_mixin_not_outputting.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 — Selectors in Mixins

Description

The mixins can contain not just properties but they can contain selectors too.

Example

The following example demonstrates the use of *mixin selectors* in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Selectors in Mixins</title>
</head>
<body>
<h2>Welcome to Tutorialspoint</h2>
<a href="http://www.tutorialspoint.com/">Tutorialspoint</a>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin() {
  &:hover {
    background: #FFC0CB;
  }
}
a {
  .mixin();
}
```

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:

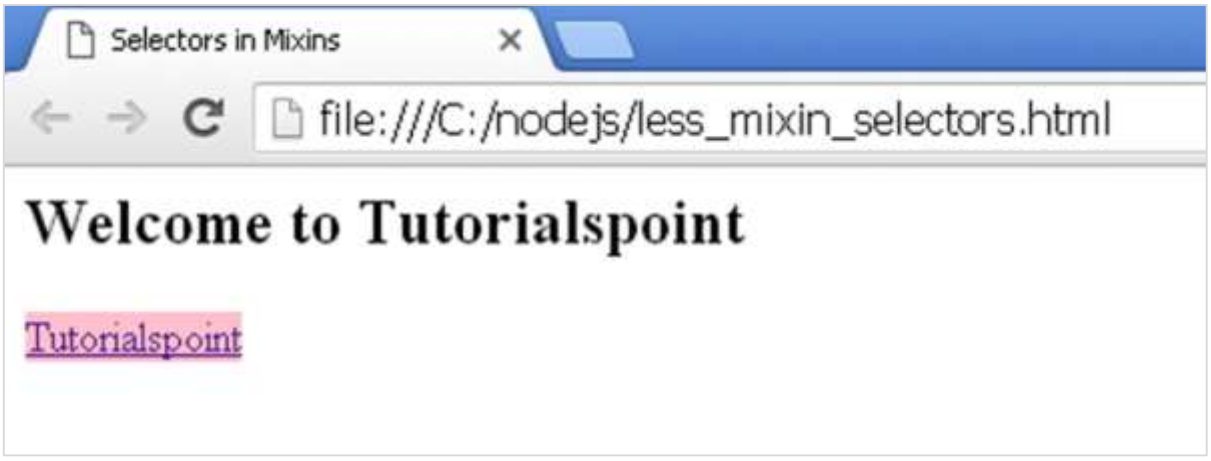
style.css

```
a:hover {
  background: #FFC0CB;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to Tutorialspoint

Tutorialspoint

LESS — Mixin Namespaces

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 *mixin namespaces* in the LESS file:

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

Next, create the *style.less* file.

style.less

```
#outer() {
  background:yellow;
  .inner {
    color: red;
  }
}
p {
  #outer > .inner;
}
```

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:

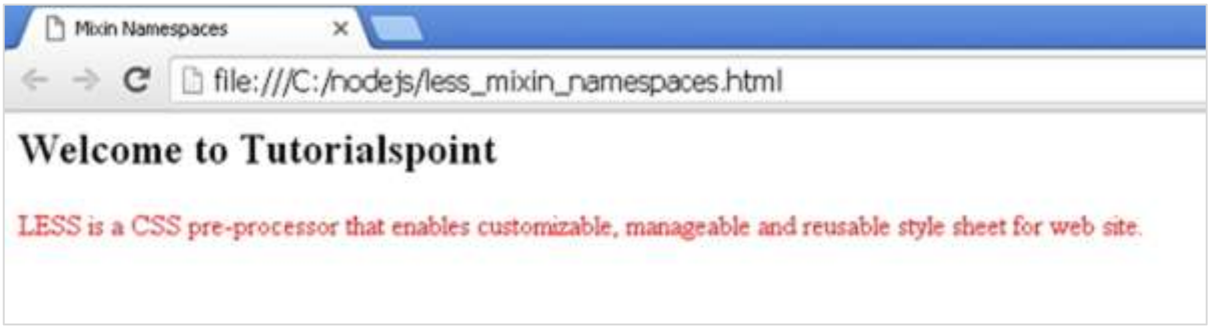
style.css

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

Output

Follow these steps to see how the above code works:

- Save the above html code in **less_mixin_namespaces.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 – Guarded Namespaces

Description

When guard is applied to namespace, a mixin defined by the namespace is used only when guard condition returns true. The **namespace guard** is similar to guard on mixins.

Example

The following example demonstrates the use of *guarded namespaces* in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Guarded Namespaces</title>
</head>
<body>


<h2>Welcome to Tutorialspoint</h2>
<p>This will paragraph be displayed red, when (@color = blue) in style.less and
when color is other than blue, then this paragraph will be default black.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
@import "http://www.tutorialspoint.com/less/lib.less";
#namespace when (@color = blue) {
  .mixin() {
   color: red;
  }
}
p{
 #namespace .mixin();
}
```

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

lib.less

```
@color: blue;
```

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

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

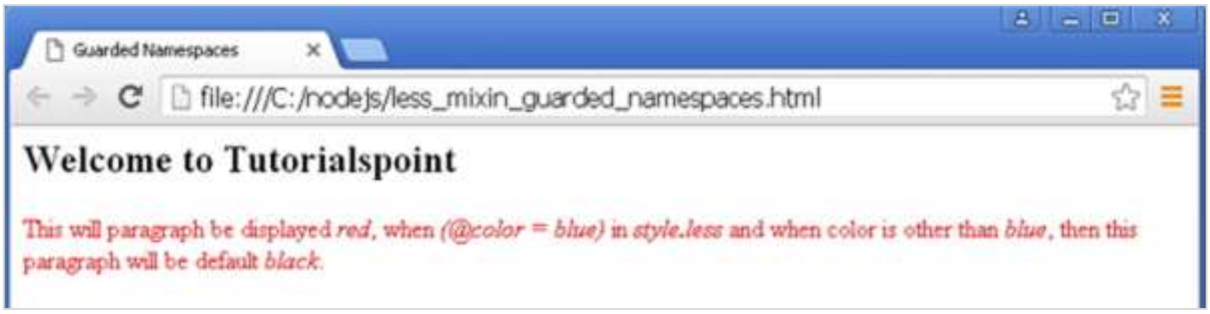
Execute the above command, it will create the *style.css* file automatically with the following code:

```
style.css
p {
  color: red;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to Tutorialspoint

This will paragraph be displayed *red*, when *(@color = blue)* in *style.less* and when color is other than *blue*, then this paragraph will be default *black*.

LESS — !important keyword

Description

The **!important** keyword is used to override the particular property. When it is placed after mixin call, it marks all inherited properties as *!important*.

The following example demonstrates the use of *!important keyword* in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>The !important keyword</title>
</head>
<body>
<h1>Welcome to Tutorialspoint</h1>
<p class="para1">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
<p class="para2">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin(){
  color: #900;
  background: #F7BE81;
}
.para1{
  .mixin();
}
.para2{
  .mixin() !important;
}
```

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:

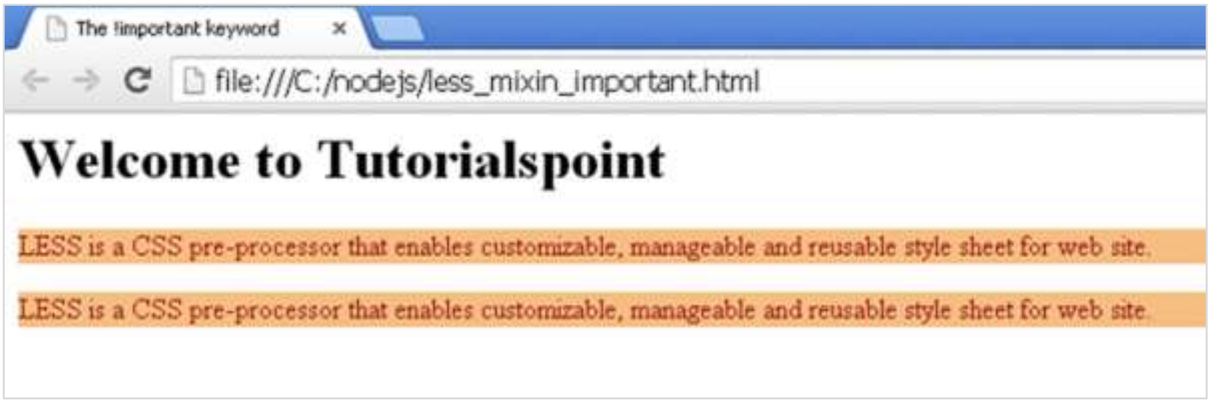
style.css

```
.para1 {
  color: #900;
  background: #F7BE81;
}
.para2 {
  color: #900 !important;
  background: #F7BE81 !important;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **less_mixin_important.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 is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.

15. LESS — Parametric Mixins

Description

Parametric mixins use one or more parameters that extend functionality of LESS by taking arguments and its properties to customize the mixin output when mixed into another block.

For instance, consider a simple LESS code snippet:

```
.border(@width; @style; @color) {
    border: @width @style @color;
}

.myheader {
    .border(2px; dashed; green);
}
```

Here we are using the parametric mixin as *.border* with three parameters — width, style and color. Using these parameters, you can customize the mixin output with the passed parameters value.

The following table describes the different types of parametric mixins along with description.

S.NO.	Types & Description
1	**Mixins with Multiple Parameters** Parameters can be separated using commas or semicolon.
2	**Named Parameters** Mixins provide parameter values instead of positions by using their names.
3	**@arguments Variable** When a mixin is called, the *@arguments* include all the passed arguments.
4	**Advanced Arguments and the @rest Variable** Mixin takes variable number of arguments by using **.....**
5	**Pattern-matching** Change the behavior of mixin by passing parameters to it.

LESS — Mixins with Multiple Parameters

Description

Parameters can be separated using *commas* or *semicolon*. Using the comma symbol, you can interpret it as mixin parameters separator or css list separator. If you use semicolon inside mixin, then it separates the arguments by semicolons and CSS lists will be having all the commas.

It includes some points on semicolons and commas as listed below:

- If you have two arguments, then the arguments will contain commas separated list. For instance, **.class1(1, 2, 3; sometext, other thing)**.
- If there are three arguments, the arguments will include only numbers such as **.class1(1, 2, 3)**.
- You can use dummy semicolon with comma separated list like **.class1(1, 2, 3;)**.
- There is comma separated default value. For instance **.class1(@color: gray, green;)**

Syntax

```
.mixin_name(@var_name1; @var_name2:some){
   //code here
}
```

Example

The following example demonstrates the use of mixin multiple parameters in the LESS file:

```
<!doctype html>
<head>
   <title>Mixin Multiple Parameters</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Mixin Multiple Parameters</h2>
   <p class="myclass">LESS enables customizable, manageable and reusable style
sheet for web site.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin(@color) {
  color: @color;
}
.mixin(@color; @padding: 2) {
  color: @color;
  padding: @padding;
}

.myclass {
  .mixin(#FE9A2E);
}
```

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:

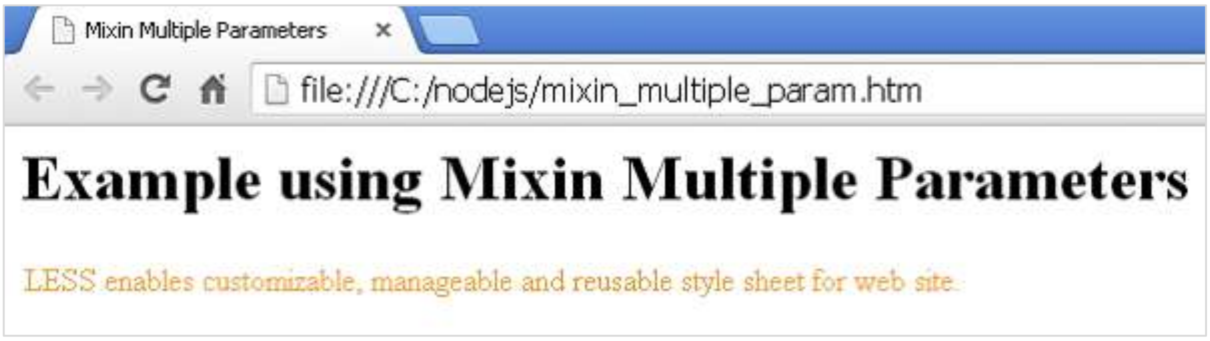
style.css

```
.myclass {
  color: #FE9A2E;
  padding: 2;
}
```

Output

Follow these steps to see how the above code works:

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


Mixin Multiple Parameters
file:///C:/nodejs/mixin_multiple_param.htm

Example using Mixin Multiple Parameters

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

LESS — Named Parameters

Description

Mixins provide parameter values instead of positions by using their names. Parameters don't have any order for placing values and they can be referenced by name. The result of named parameters is easier to read and provides clear codes.

Example

The following example demonstrates the use of named parameters in the LESS file:

```
<!doctype html>
<head>
   <title>Named Parameters</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Named Parameters</h2>
   <p class="class1">Hello World...</p>
   <p class="class2">Welcome to Tutorialspoint...</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin(@color: black; @fontSize: 10px) {
  color: @color;
  font-size: @fontSize;
}
.class1 {
  .mixin(@fontSize: 20px; @color: #F5A9D0);
}
.class2 {
  .mixin(#F79F81; @fontSize: 20px);
}
```

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:

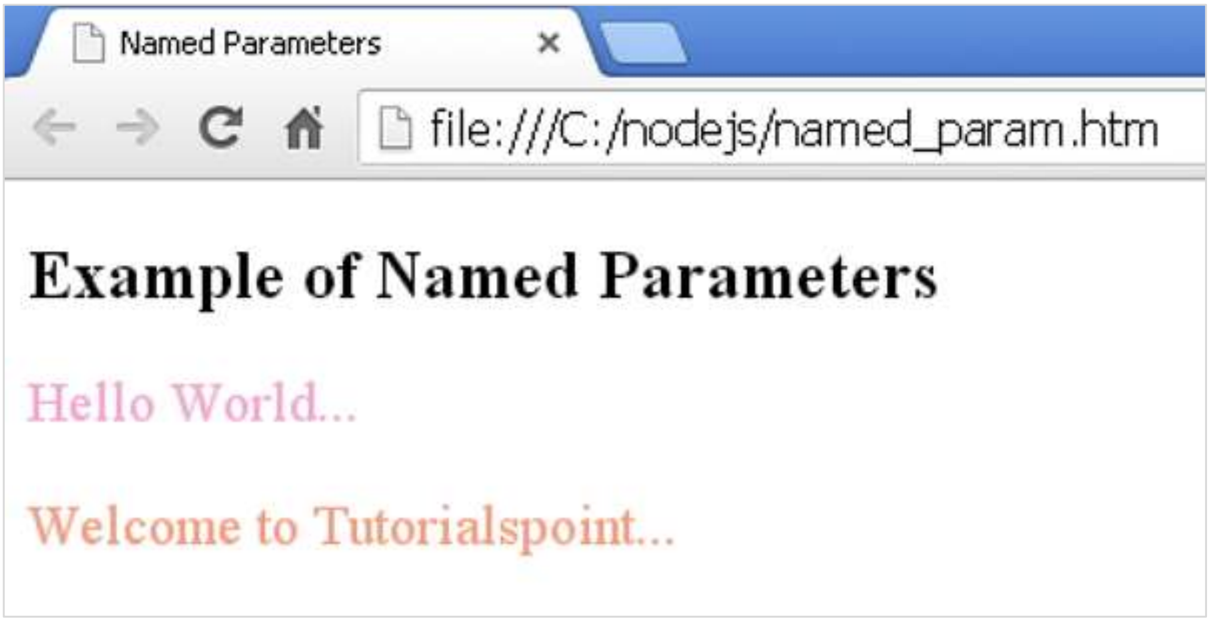
style.css

```
.class1 {
  color: #F5A9D0;
  font-size: 20px;
}
.class2 {
  color: #F79F81;
  font-size: 20px;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Named Parameters

Hello World...

Welcome to Tutorialspoint...

LESS — @arguments Variable

Description

When a mixin is called, the *@arguments* include all the passed arguments. The *@arguments* variable is useful when you don't want to work with individual parameters.

Example

The following example demonstrates the use of named parameters in the LESS file:

```
<!doctype html>
<head>
   <title>@arguments Variable</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of @arguments Variable</h2>
   <p class="myclass">Welcome to Tutorialspoint...</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.box-shadow(@x: 0; @y: 0; @height: 3px; @width: 3px) {
  -webkit-box-shadow: @arguments;
     -moz-box-shadow: @arguments;
          box-shadow: @arguments;
}
.myclass {
  .box-shadow(2px; 2px);
}
```

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:

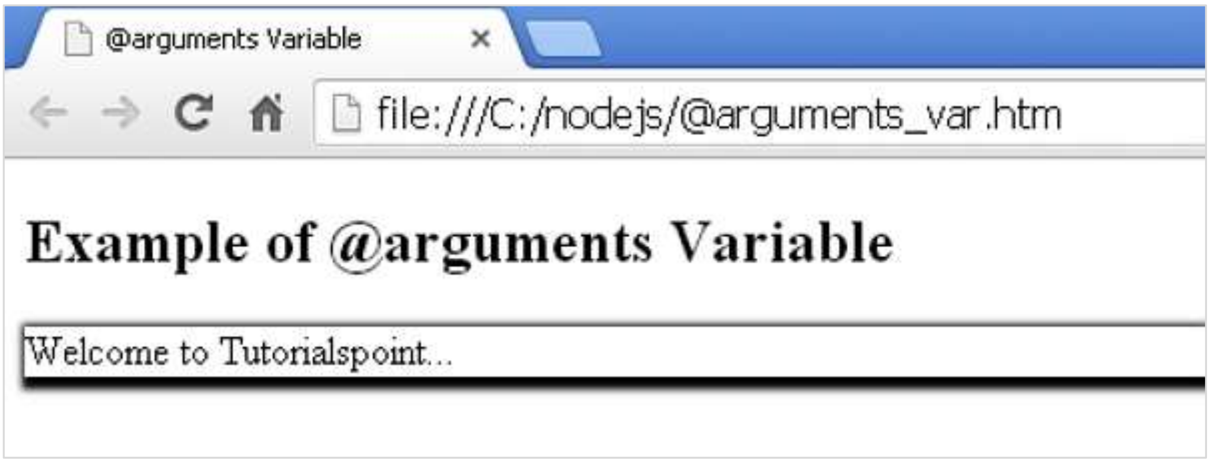
style.css

```
.myclass {
  -webkit-box-shadow: 2px 2px 3px 3px;
  -moz-box-shadow: 2px 2px 3px 3px;
  box-shadow: 2px 2px 3px 3px;
}
```

Output

Follow these steps to see how the above code works:

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



Example of @arguments Variable

Welcome to Tutorialspoint...

Less Advanced Arguments and @rest Variable

Description

Mixin takes variable number of arguments by using **....** You can assign arguments to the variable by placing **...** after the variable name.

The following program shows simple formats of using arguments:

```
.mixin(...) {        // it matches arguments from 0-n
.mixin() {           // it matches exactly 0 arguments
.mixin(@x: 1) {      // it matches arguments from 0-1
.mixin(@x: 1; ...) { // it matches arguments from 0-n
.mixin(@x; ...) {
```

You can use the **@rest** variable in the code as:

```
.mixin(@x; @rest...) {
   // after the variable @a, the @rest is bound to arguments
   // @arguments is bound to all arguments
}
```

LESS — Pattern Matching

Description

You can change the behavior of mixin by passing parameters to it.

Consider one simple LESS code snippet:

```
.mixin(@a; @color) { ... }

.line {
  .mixin(@color-new; #888);
}
```

You can use different values for @color-new to make different mixin behaviors as shown in the code given below.

```
.mixin(dark; @color) {
  color: darken(@color, 15%);
}
.mixin(light; @color) {
  color: lighten(@color, 15%);
}

@color-new: dark;

.line {
  .mixin(@color-new; #FF0000);
}
```

If you set the value of the *@color-new* to dark, then it will display the result in darker color as mixin definition matches the *dark* as first argument.

Example

The following example demonstrates the use of pattern matching in the LESS file:

```
<!doctype html>
<head>
   <title>Pattern Matching</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Pattern Matching</h2>
   <p class="myclass">Welcome to Tutorialspoint...</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin(dark; @color) {
  color: darken(@color, 15%);
}
.mixin(light; @color) {
  color: lighten(@color, 15%);
}

@color-new: dark;

.myclass {
  .mixin(@color-new; #FF0000);
}
```

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:

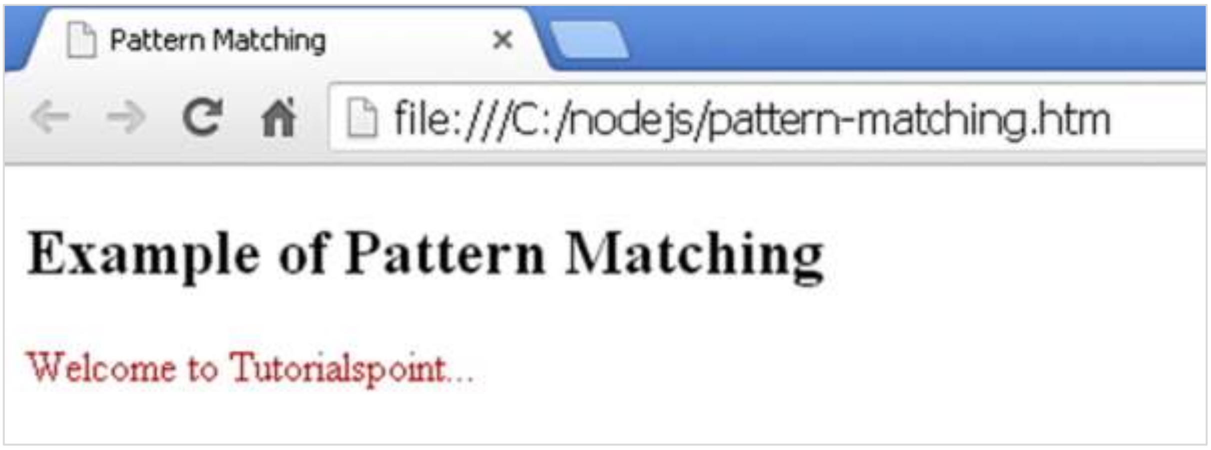
style.css

```
.myclass {
  color: #b30000;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Pattern Matching

Welcome to Tutorialspoint...

16. Less — Mixins as Functions

In this chapter, we will understand the importance of **Mixins as Functions**. Like functions, mixins can be nested, can accept parameters, and return values too.

The following table demonstrates the use of *mixins as functions* in details.

S.NO.	Mixins usage & Description
1	**Mixin scope** Mixins consist of variables; these can be used in caller's scope and are visible.
2	**Mixin and return values** Mixins are similar to functions and the variables that are defined in a mixin will behave as the return values.
3	**Mixin inside another mixin** Whenever a mixin is defined inside another mixin, it can be used as return value too.

LESS — Mixin scope

Description

Mixins consisting of variables are visible and can be used in caller's scope. But there is one exception, if the caller contains a variable with the same name, then that variable is not copied into the caller's scope. Only the variables inside the caller's scope are protected and the inherited variables are overridden.

Example

The following example demonstrates the use of *mixin scope* in the LESS file:

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

```
.mixin() {
  @bgcolor: #C0C0C0;
}
.myclass{
  .mixin();
  background-color: @bgcolor;
}
```

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:

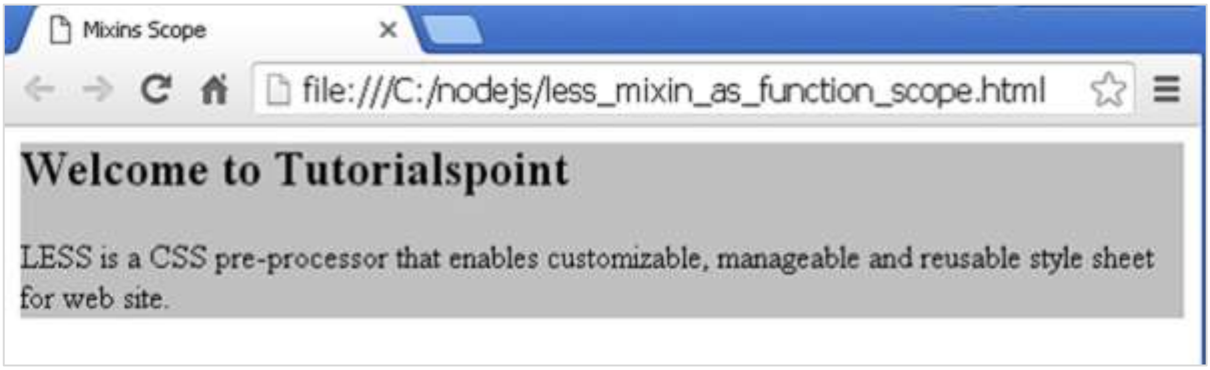
style.css

```
.myclass {
  background-color: #C0C0C0;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **less_mixin_as_function_scope.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.

Variables cannot be overridden in the caller's scope when they are defined directly in it. For more information click here.

LESS — Mixin scope

Description

Variables cannot be overridden in the caller's scope when they are defined directly in it. However, variables are not protected and will be overridden when it is defined in callers parent scope.

Example

The following example demonstrates the use of *mixin scope* in the LESS file:

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

```
@val: 20px; // callers parent scope - no protection
.mixin() {
  @val: 10px;
  @definedOnlyInMixin: 10px;
}
.myclass {
  padding-left: @val * @definedOnlyInMixin;
  .mixin();
}
```

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:

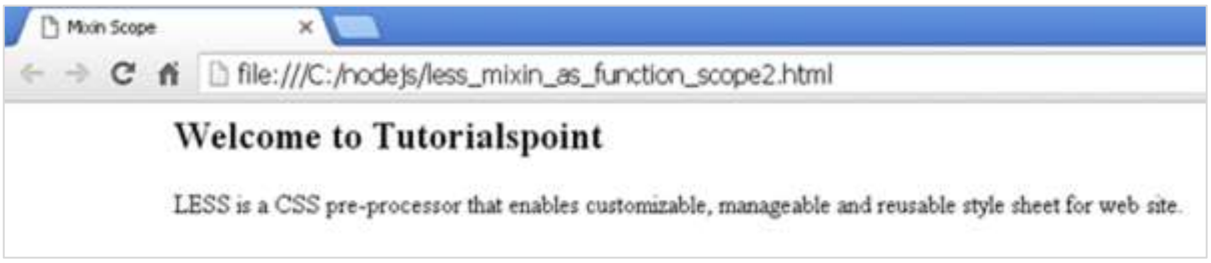
style.css

```
.myclass {
  padding-left: 100px;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **less_mixin_as_function_scope2.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 — Mixin & return values

Description

Mixins are similar to functions and the variables that are defined in a mixin will behave as its return values.

Example

The following example demonstrates the use of *mixin & return values* in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Mixins & return values</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

```
.padding(@x, @y) {
  @padding: ((@x + @y) / 2);
}

.myclass{
  .padding(80px, 120px);  // call to the mixin
  padding-left: @padding; //  returns the value
}
```

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:

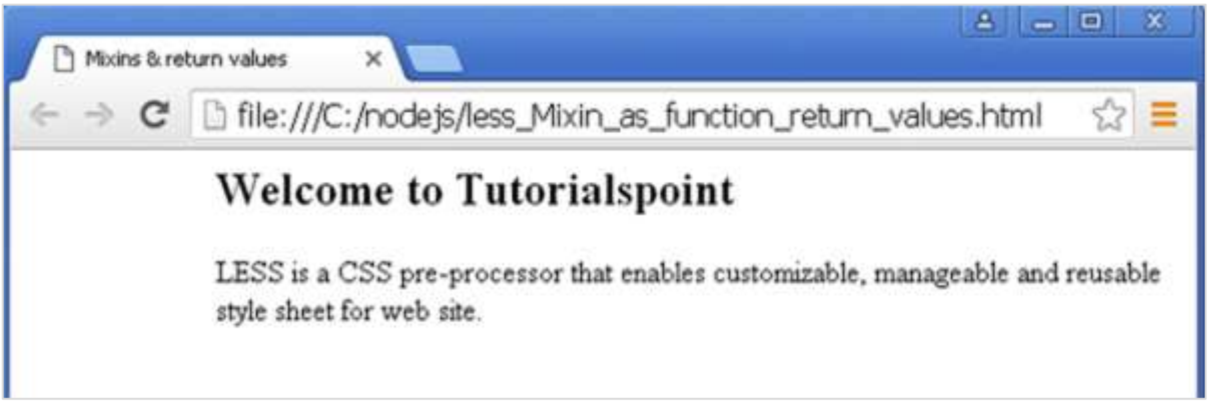
style.css

```
.myclass {
  padding-left: 100px;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **less_Mixin_as_function_return_values.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 — Mixin inside mixin

Description

Whenever a mixin is defined inside another mixin, it can be used as return value too.

Example

The following example demonstrates the use of *mixin inside mixin* in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Mixin inside mixin</title>
```

```
</head>
<body>
<h1>Welcome to Tutorialspoint</h1>
<div class="myclass">
 <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

```
.outerMixin(@value) {
  .nestedMixin() {
    font-size: @value;
  }
}
.myclass {
  .outerMixin(30);
  .nestedMixin();
}
```

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:

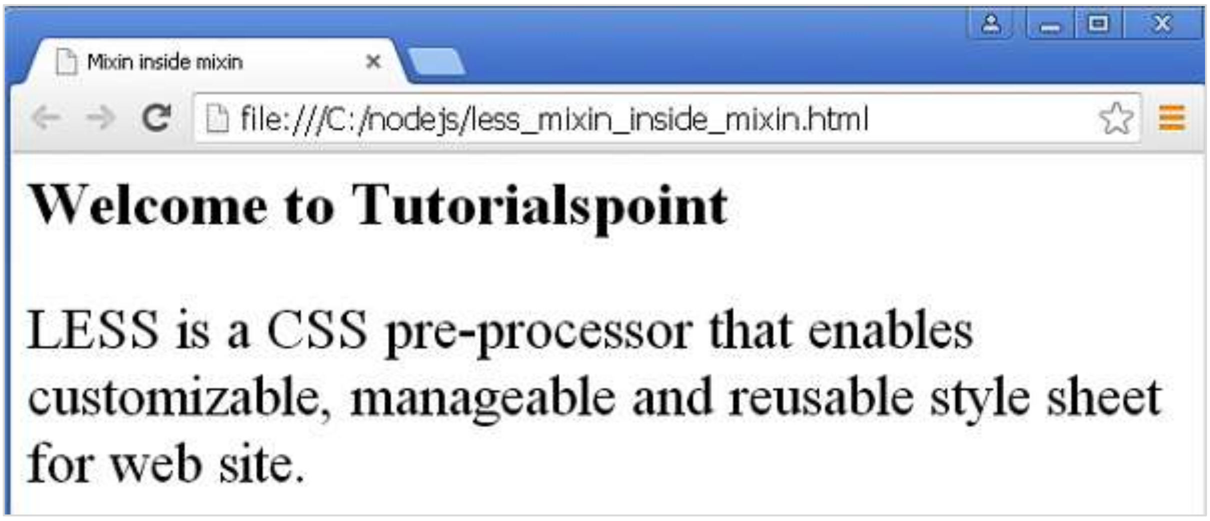
style.css

```
.myclass {
  font-size: 30;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **less_mixin_inside_mixin.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.

17. LESS — Passing Rulesets to Mixins

Description

Detached ruleset contains rulesets such as properties, nested rulesets, variables declaration, mixins, etc. It is stored in a variable and included in another structure; all the properties of the ruleset get copied to that structure.

Example

The following example shows how to pass a ruleset to mixin in the LESS file:

passing_ruleset.htm

```
<!doctype html>
<head>
     <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="cont">
   <h2>Welcome to TutorialsPoint</h2>
   <p>The largest Tutorials Library on the web.</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
@detached-ruleset: {
    .mixin() {
        font-family: "Comic Sans MS";
        background-color: #AA86EE;
    }
};

.cont {
    @detached-ruleset();
    .mixin();
}
```

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:

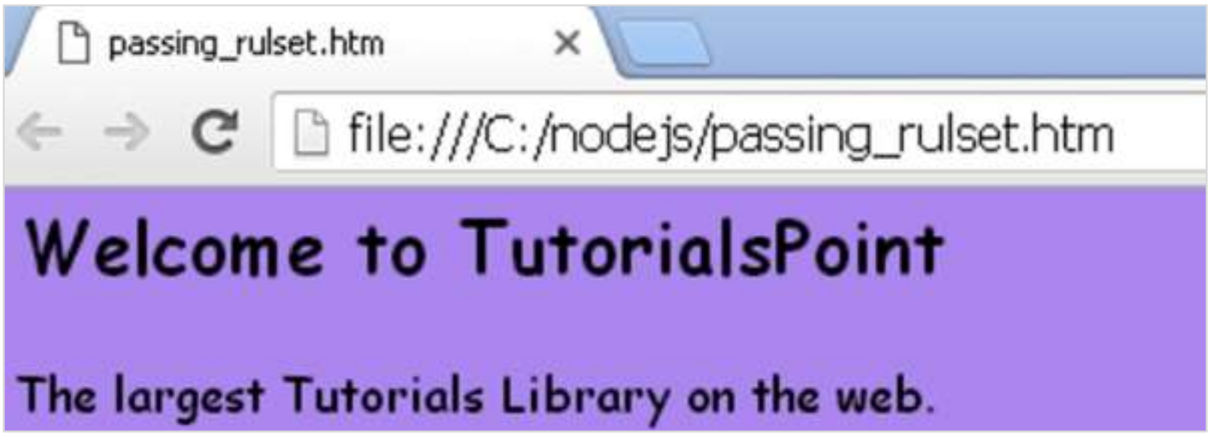
style.css

```
.cont {
  font-family: "Comic Sans MS";
  background-color: #AA86EE;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

Scoping

All variables and mixins in detached ruleset are available wherever the ruleset called or defined. Otherwise, both the caller and the definition scopes are available by default. The declaration scope takes the priority when both scopes contain same mixin or variable. Detached ruleset body is defined in the declaration scope. It does not change its scope after the detached ruleset is copied from one variable to another.

The following table lists all the types of scope:

S.NO.	Types & Description
1	**Definition and Caller Scope Visibility** Variables and mixins are defined inside the detached ruleset.
2	**Referencing Won't Modify Detached Ruleset Scope** Just giving the references, the ruleset does not access to any new scopes.
3	**Unlocking Will Modify Detached Ruleset Scope** The detached ruleset can access to scope by being imported into it.

LESS — Definition and Caller Scope Visibility

Description

Variables and mixins are defined inside the detached ruleset.

Example

The following example demonstrates the use of variable and mixin defined inside the ruleset in the LESS file:

passing_ruleset.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="cont">
   <h2>Welcome to TutorialsPoint</h2>
   <p>The largest Tutorials Library on the web.</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
@detached-ruleset: {
  background-color: @caller-variable;
  .caller-mixin();
};

.cont {
  @detached-ruleset();

  @caller-variable: #AA86EE;
  .caller-mixin() {
    font-style:italic;
  }
}
```

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:

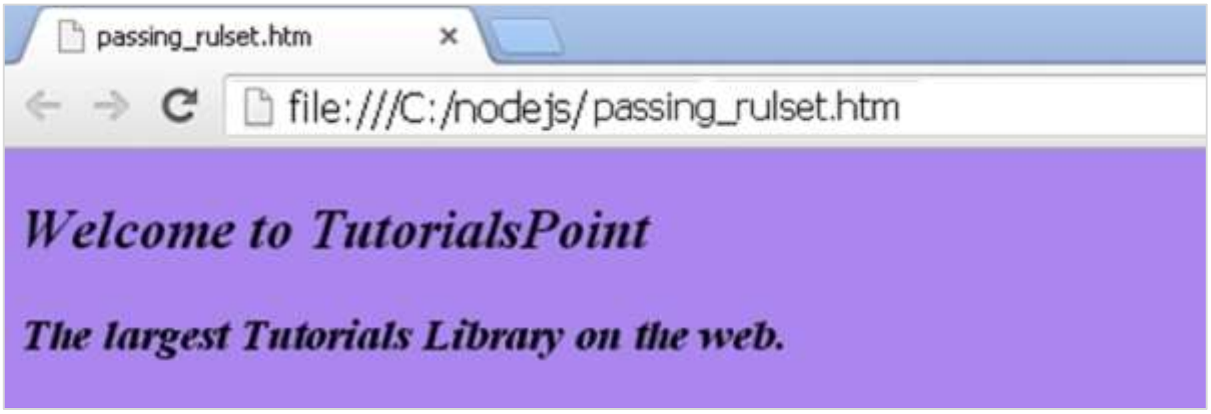
style.css

```
.cont {
  background-color: #AA86EE;
  font-style: italic;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

LESS — Referencing Won't Modify Detached Ruleset Scope

Description

By being referenced, the ruleset does not gain access to any new scopes.

Example

The following example demonstrates the use of giving references. However, this doesn't modify the detached ruleset scope in the LESS file:

passing_ruleset.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="cont">
   <h2>Welcome to TutorialsPoint</h2>
   <h3>The largest Tutorials Library on the web.</h3>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

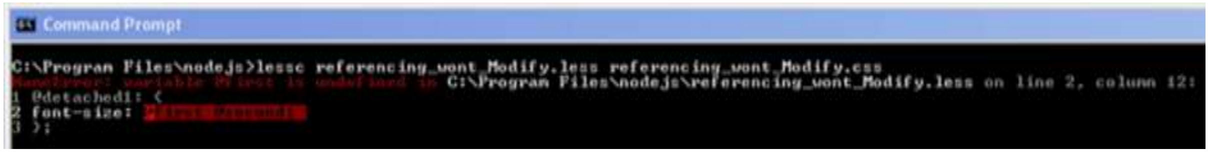
```
@detached1: {
font-size: @first @second;
};

.first {
  @first: 25px;
  .second {
    @detached2: @detached1;
    @second: 30px;
  }
}
.cont {
  .first > .second();
  @detached2();
}
```

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

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

Execute the above command, you will get the following error in the cmd:



```
C:\Program Files\nodejs>lessc referencing_wont_Modify.less referencing_wont_Modify.css
NameError: variable @first is undefined in C:\Program Files\nodejs\referencing_wont_Modify.less on line 2, column 12:
1 @detached1: {
2 font-size: [illegible]
3 };
```

LESS — Unlocking Will Modify Detached Ruleset Scope

Description

The detached ruleset can gain access to scope by being imported into it.

Example

The following example demonstrates unlocking. This will modify the detached ruleset scope in the LESS file:

passing_ruleset.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
```

```
<body>
<div class="cont">
   <h2>Welcome to TutorialsPoint</h2>
   <p>The largest Tutorials Library on the web.</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
#id {
  .style() {
    @detached: { font-style: @font; };
  }
}

.container() {
  @font: italic;
  #id > .style();
}

.cont {
  .container();
   @detached();
}
```

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:

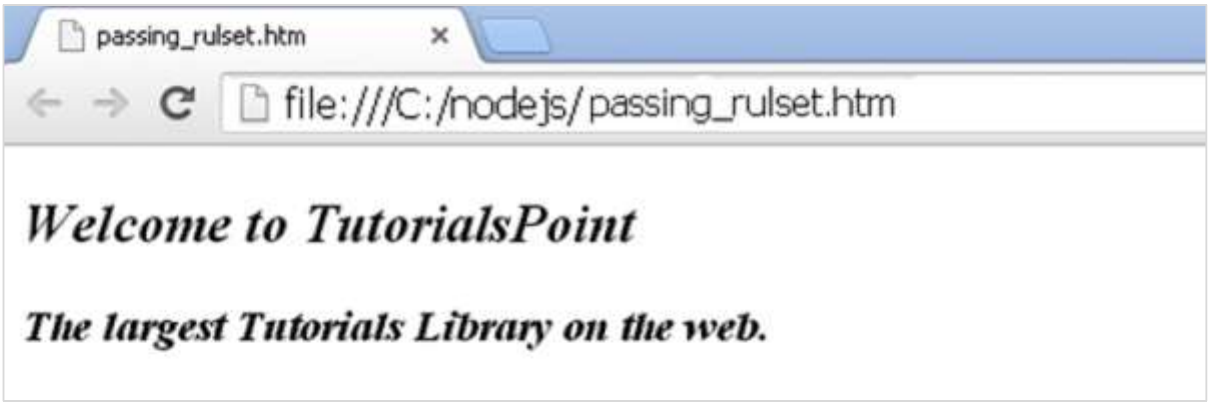
style.css

```
.cont {
  font-style: italic;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

18. Less — Import Directives

Description

The **@import** directive is used to import the files in the code. It spreads the LESS code over different files and allows to maintain the structure of code easily. You can put the *@import* statements anywhere in the code.

For instance, you can import the file by using **@import** keyword as *@import "file_name.less"*.

File Extensions

You can use the *@import* statements depending on the different file extensions such as:

- If you are using *.css* extension, then it will be considered as CSS and the *@import* statement remains as it is.
- If it contains any other extension, then it will be considered as LESS and will be imported.
- If there is no LESS extension, then it will appended and included as imported LESS file.

```
@import "style";      // imports the style.less
@import "style.less"; // imports the style.less
@import "style.php";  // imports the style.php as a less file
@import "style.css";  // it will kept the statement as it is
```

Example

The following example demonstrates the use of variable in the SCSS file:

```
<!doctype html>
<head>
   <title>Import Directives</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Import Directives</h2>
   <p class="myline">Welcome to Tutorialspoint...</p>
</body>
</html>
```

Next, create the *import_dir.less* file.

import_dir.less

```
.myline {
 font-size: 20px;
}
```

Now, create the *style.less* file.

style.less

```
@import "http://www.tutorialspoint.com/less/import_dir.less";
.myline {
  color:#FF0000;
}
```

The *import_dir.less* file will get imported into *style.less* file from the path http://www.tutorialspoint.com/less/import_dir.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:

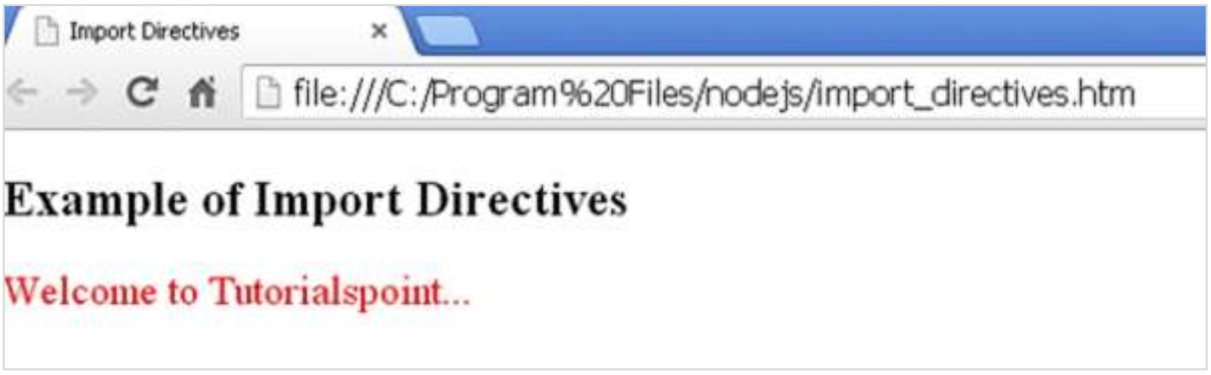
style.css

```
.myline {
  font-size: 20px;
}
.myline {
  color: #FF0000;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Import Directives

Welcome to Tutorialspoint...

19. LESS — Import Options

In this chapter, we will understand the importance of **Import Options** in LESS. LESS offers the **@import** statement that allows the style sheets to import both LESS and CSS style sheets.

The following tables lists the import directives that will be implemented in the import statements.

S.NO.	Import options & Description
1	**reference** It uses a LESS file only as reference and will not output it.
2	**inline** It enables you to copy your CSS into the output without being processed.
3	**less** It will treat the imported file as the regular LESS file, despite whatever may be the file extension.
4	**css** It will treat the imported file as the regular CSS file, despite whatever may be the file extension.
5	**once** It will import the file only one time.
6	**multiple** It will import the file multiple times.
7	**optional** It continues compiling even though the file to import is not found.

More than one keyword is allowed to use in the **@import** statement, however you have to use commas to seperate the keywords.

For instance:

```
@import (less, optional) "custom.css";
```

LESS — Import Options Reference Keyword

Description

The **@import (reference)** is used to import external files but it will not add imported styles to compiled CSS file. This was released in *version 1.5.0*.

Example

The following example demonstrates the use of *reference* keyword in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Import Options Reference</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 *style.less* file.

style.less

```
@import (reference) "http://www.tutorialspoint.com/less/import_reference.less";
p {
    .style1;
}
```

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

import_reference.less

```
.style1 {
    color: #A0A0A0;
    font-family: "Comic Sans MS";
    font-size: 20px;
}
```

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:

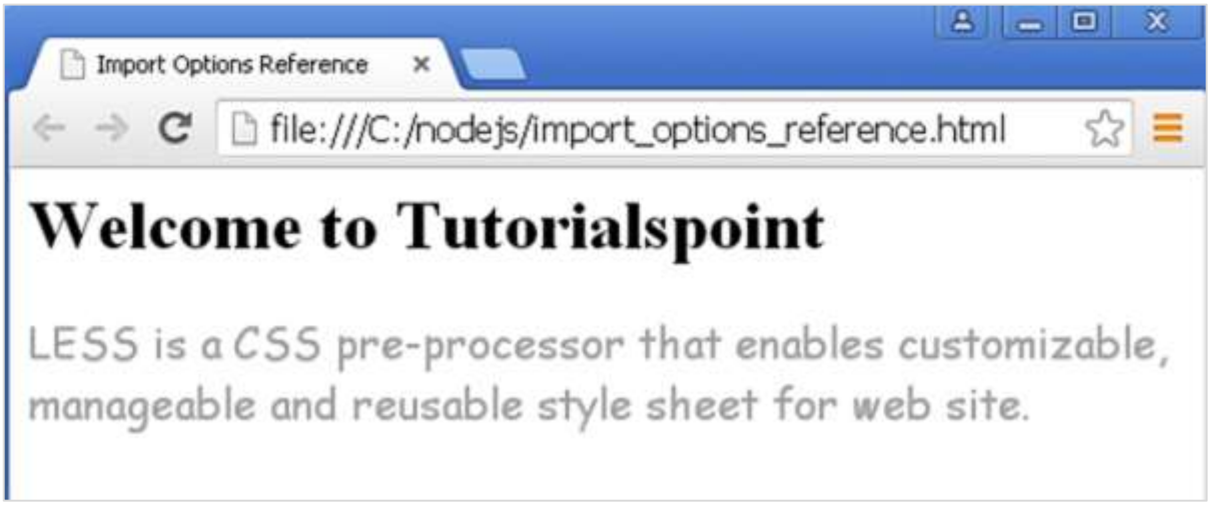
style.css

```
p {
    color: #A0A0A0;
    font-family: "Comic Sans MS";
    font-size: 20px;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **import_options_reference.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.

The **reference** has the **extend** method that pulls in a new selector at the place where you refer the **@import** statement and marks it as *not referenced*. For more information, click here.

LESS – Import Options Extend

Description

When a selector is extended, a new selector is pulled in at the place where the **@import** statement is referred and only the new selector is marked as *not referenced*. We can use **extend** to avoid this redundancy and pull in only specific targeted styles.

Example

The following example demonstrates the use of the **extend** method in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Import Options Extend</title>
</head>
<body>
<h1>Welcome to Tutorialspoint</h1>
<p class="para_1">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
<p class="para_2">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
@import (reference)
"http://www.tutorialspoint.com/less/import_options_extend.less";
.para_1 {
    &:extend(.style1);
    background-color: #81F7F3;
}
.para_2 {
    &:extend(.style1);
    background-color: #F6E3CE;
}
```

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

import_options_extend.less

```
.style1 {
    color: #A0A0A0;
    font-family: "Comic Sans MS";
    font-size: 20px;
}
```

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:

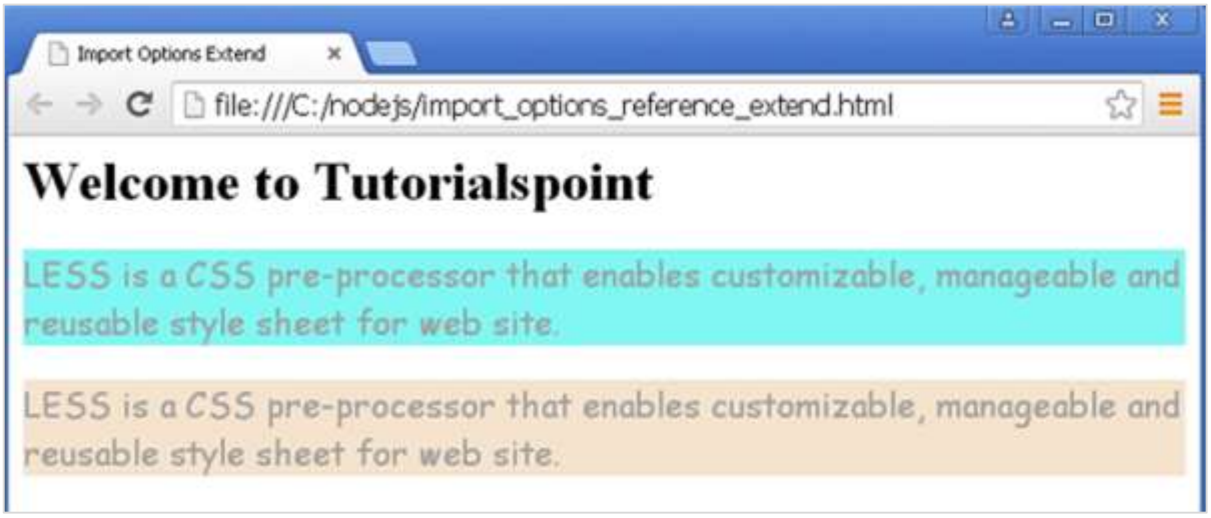
style.css

```
.para_1,
.para_2 {
  color: #A0A0A0;
  font-family: "Comic Sans MS";
  font-size: 20px;
}
.para_1 {
  background-color: #81F7F3;
}
.para_2 {
  background-color: #F6E3CE;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **import_options_reference_extend.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 is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.

LESS — Import Options Inline Keyword

Description

The **@import (inline)** statement will copy your CSS into the output CSS file without processing it. This is useful when the CSS file is not LESS compatible. Although LESS supports most standards CSS, comments are not supported in some places and without modifying the CSS, it will not support all known CSS hacks. Even though **@import (inline)** will not process the CSS, it will ensure that all your CSS will be in one file. This was released in *version 1.5.0*.

Example

The following example demonstrates the use of *reference* keyword in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Import Option Inline</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 *style.less* file.

style.less

```
@import (inline) "http://www.tutorialspoint.com/less/import_inline.css";
p {
  color:red;
}
```

The following code will import the *import_inline.css* file into *style.less* from the path http://www.tutorialspoint.com/less/import_inline.css with the following code:

import_inline.css

```
.style {
    font-family: "Comic Sans MS";
    font-size: 20px;
}
```

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:

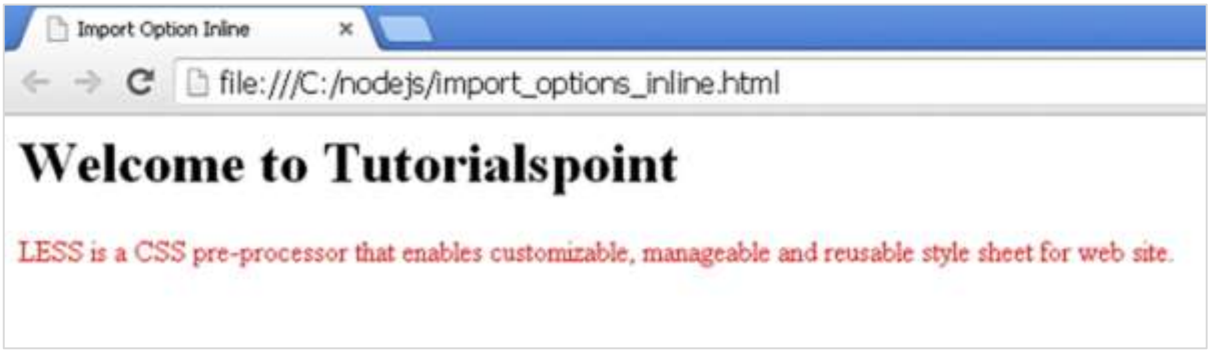
style.css

```
.style {
    font-family: "Comic Sans MS";
    font-size: 20px;
}
p {
  color: red;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **import_options_inline.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.

If you try to use the ***.style*** class inside the ***p*** tag in ***style.less***, it will throw an *undefined error*.

Less – Import Options Less Keyword

Description

The **@import (less)** will import the file as LESS file, regardless of whatever may be the file extension. This was released in *version 1.4.0*.

Example

The following example demonstrates the use of *less* keyword in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Import Options Less</title>
</head>
<body>
<h1>Welcome to Tutorialspoint</h1>
<p class="para_1">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
<p class="para_2">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
@import (less) "http://www.tutorialspoint.com/less/less.txt";
.para_1 {
    color: red;
    .style;
}
.para_2 {
    color: blue;
}
```

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

less.txt

```
.style {
  font-family: "Comic Sans MS";
  font-size: 20px;
}
```

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:

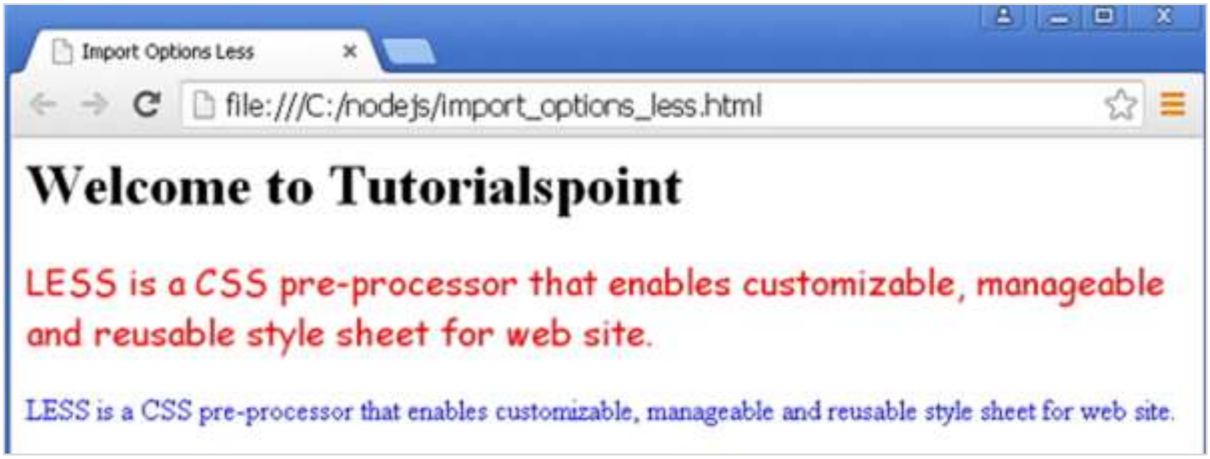
style.css

```
.style {
  font-family: "Comic Sans MS";
  font-size: 20px;
}
.para_1 {
  color: red;
  font-family: "Comic Sans MS";
  font-size: 20px;
}
.para_2 {
  color: blue;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **import_options_less.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 is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.

LESS – Import Options CSS Keyword

Description

The **@import (css) keyword** will import the file as regular CSS, regardless of file extension. This was released in *version 1.4.0*.

Example

The following example demonstrates the use of *css* keyword in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Import Options CSS</title>
</head>
<body>
<h1>Welcome to Tutorialspoint</h1>
<p class="para_1">LESS is a CSS pre-processor that enables customizable,
manageable and reusable style sheet for web site.</p>
<p class="para_2">LESS is a CSS pre-processor that enables customizable,
manageable and reusable style sheet for web site.</p>
```

```
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
@import (css) "http://www.tutorialspoint.com/less/css.txt";
.para_1 {
    color: green;
    .my_css;
}
.para_2 {
    color: blue;
}
```

The following code will import the *css.txt* file into *style.less* from the http://www.tutorialspoint.com/less/css.txt path:

css.txt

```
.my_css {
    font-family: "Comic Sans MS";
    font-size: 20px;
}
```

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:

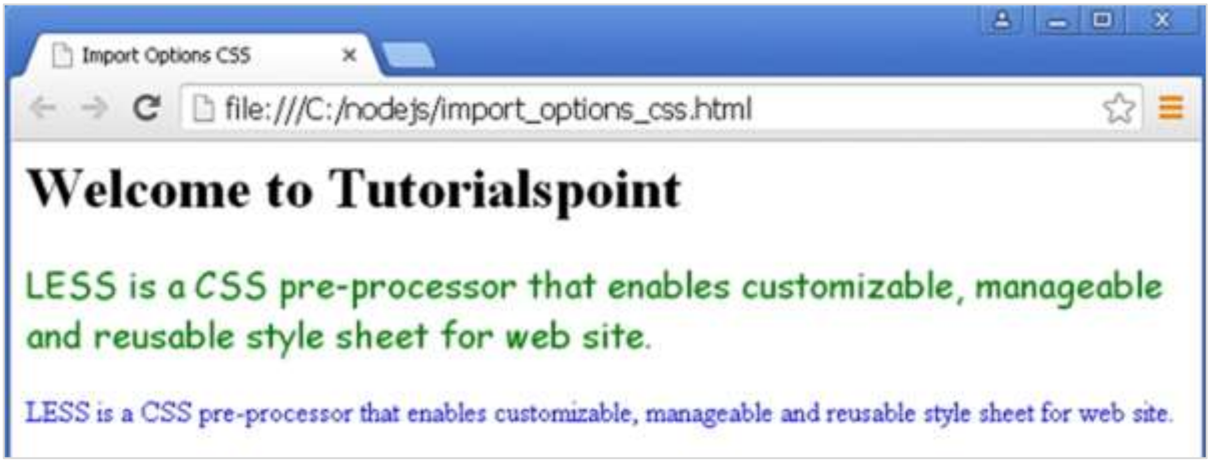
style.css

```
.my_css {
  font-family: "Comic Sans MS";
  font-size: 20px;
}
.para_1 {
  color: green;
  font-family: "Comic Sans MS";
  font-size: 20px;
}
.para_2 {
  color: blue;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **import_options_css.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 is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.

LESS — Import Options Once Keyword

Description

The **@import (once)** keyword ensures that the file is imported only once and any following import statements will be neglected for that file. This is the default behavior of the **@import** statements. This was released in *version 1.4.0*.

Example

The below example demonstrates the use of *once* keyword in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Import Options Once</title>
</head>
<body>
```

```
<h1>Welcome to Tutorialspoint</h1>
<p class="para_1">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
<p class="para_2">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Now, create the *style.less* file.

style.less

```
@import (once) "http://www.tutorialspoint.com/less/once.less";
@import (once) "http://www.tutorialspoint.com/less/once.less"; // this statement will be ignored
.para_1 {
    color: red;
    .style;
}
.para_2 {
    color: blue;
}
```

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

once.less

```
.style {
    font-family: "Comic Sans MS";
    font-size: 20px;
}
```

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:

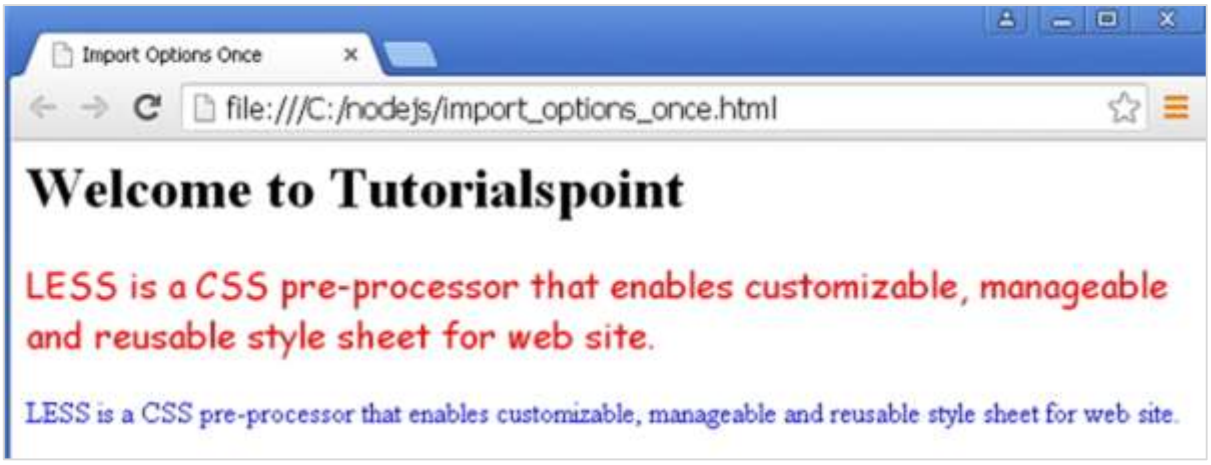
style.css

```
.style {
  font-family: "Comic Sans MS";
  font-size: 20px;
}
.para_1 {
  color: red;
  font-family: "Comic Sans MS";
  font-size: 20px;
}
.para_2 {
  color: blue;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **import_options_once.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 is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.

LESS — Import Options Multiple Keyword

Description

The **@import (multiple)** keyword enables you to import multiple files with the same name. This works exactly opposite to *once*. This was released in *version 1.4.0*.

Example

The following example demonstrates the use of *multiple* keyword in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Import Options Multiple</title>
</head>
<body>
<h1>Welcome to Tutorialspoint</h1>
<p class="para_1">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
<p class="para_2">LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
@import (multiple) "http://www.tutorialspoint.com/less/multiple.less";
@import (multiple) "http://www.tutorialspoint.com/less/multiple.less";
.para_1 {
    color: red;
}
.para_2 {
    color: blue;
}
```

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

multiple.less

```
.style {
    font-family: "Comic Sans MS";
    font-size: 20px;
}
```

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:

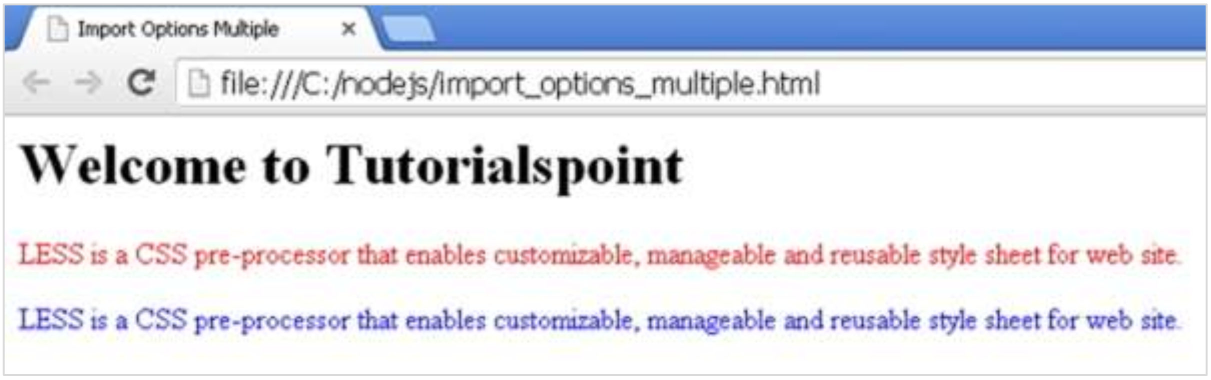
style.css

```
.style {
  font-family: "Comic Sans MS";
  font-size: 20px;
}
.style {
  font-family: "Comic Sans MS";
  font-size: 20px;
}
.para_1 {
  color: red;
}
.para_2 {
  color: blue;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **import_options_multiple.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 is a CSS pre-processor that enables customizable, manageable and reusable style sheet for web site.

LESS — Import Options Optional Keyword

Description

The **optional** keyword allows you to import a file whenever a file does not exist. If the file to be imported does not exist and the **optional** keyword is not used then, LESS throws *FileError* error and stops compiling. This functionality was released in *version 2.3.0*.

Example

The following example demonstrates the use of *optional* keyword in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Import Options Optional</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 *style.less* file.

style.less

```
@import (optional) "fileNotExist.css";
p{
  color: red;
}
```

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:

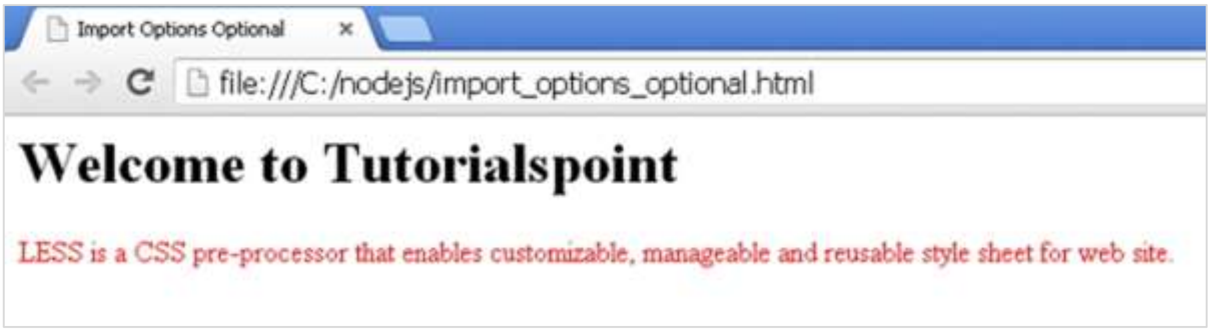
style.css

```
@import "fileNotExist.css";
p {
  color: red;
}
```

Output

Follow these steps to see how the above code works:

- Save the above html code in **import_options_optional.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.

Even though the *fileNotExist.css* file does not exist, LESS will continue compiling and will create the style.css file automatically.

20. LESS — Mixin Guards

Description

If you want to match simple values or number of arguments on expressions, then you can make use of guards. It is associated with mixin declaration and includes condition that is attached to a mixin. Each mixin will have one or more guards which are separated by comma; a guard must be enclosed within parentheses. LESS uses guarded mixins instead of **if/else** statements and performs calculations to specify matched mixin.

The following table describes the different types of mixins guards along with description.

S.NO.	Types & Description
1	**Guard Comparison Operators** You can use the comparison operator (=) to compare numbers, strings, identifiers, etc.
2	**Guard Logical Operators** You can use the *and* keyword to work around logical operators with guards.
3	**Type Checking Functions** It contains the built-in functions to determine the value types for matching mixins.
4	**Conditional Mixins** LESS uses the *default* function to match mixin with other mixing matches.

Example

The following example demonstrates the use of mixin guards in the LESS file:

```
<!doctype html>
<head>
   <title>Mixin Guards</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Mixin Guards</h2>
   <p class="class1">Hello World...</p>
   <p class="class2">Welcome to Tutorialspoint...</p>
</body>
</html>
```

Now, create the *style.less* file.

style.less

```
.mixin (@a) when (lightness(@a) >= 50%) {
   font-size: 14px;
}
.mixin (@a) when (lightness(@a) < 50%) {
   font-size: 16px;
}
.mixin (@a) {
   color: @a;
}
.class1 {
   .mixin(#FF0000)
}
.class2 {
   .mixin(#555)
}
```

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:

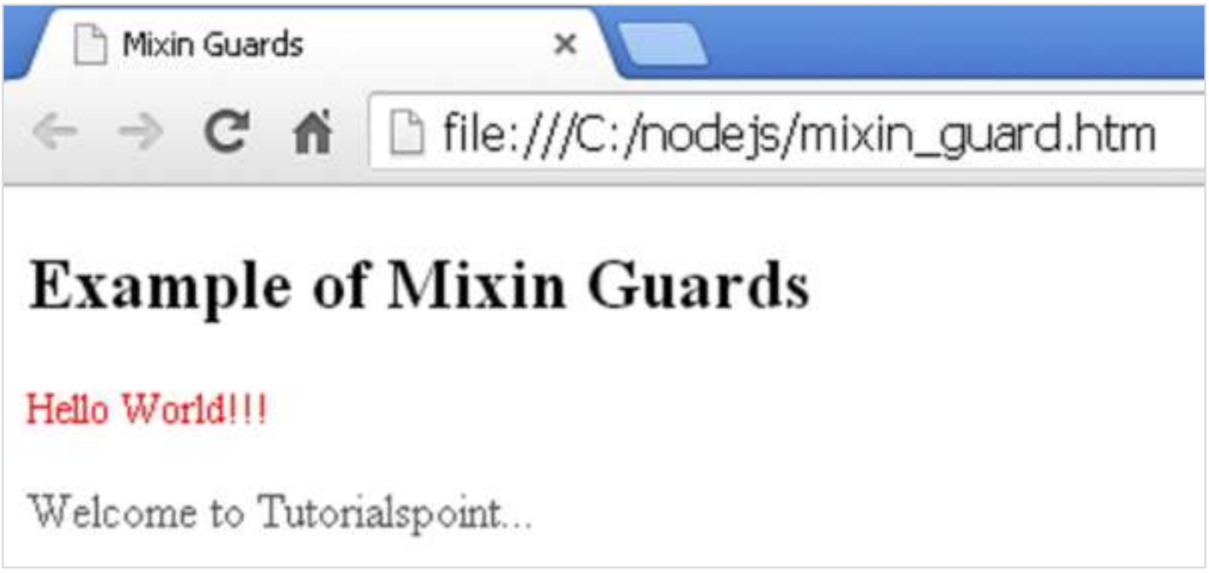
style.css

```
.class1 {
  font-size: 14px;
  color: #FF0000;
}
.class2 {
  font-size: 16px;
  color: #555;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Mixin Guards

Hello World!!!

Welcome to Tutorialspoint...

LESS – Guard Comparison Operators

Description

LESS contains five guard comparison operators: <, >, <=, >= and =. You can use the comparison operator (=) to compare numbers, strings, identifiers, etc. and the remaining operators can be used only with numbers.

Example

The following example demonstrates the use of guard comparison operators in the LESS file:

```
<!doctype html>
<head>
   <title>Guard Comparison Operators</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
```

```
    <h2>Example of Guard Comparison Operators</h2>
    <p class="myclass">Hello World!!!Welcome to Tutorialspoint...</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin (@a) when (@a = 20px){
color:red;
}

.mixin (@a) when (@a < 20px){
color:blue;
}

.mixin (@a) {
  font-size: @a;
}

.myclass { .mixin(20px) }
```

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:

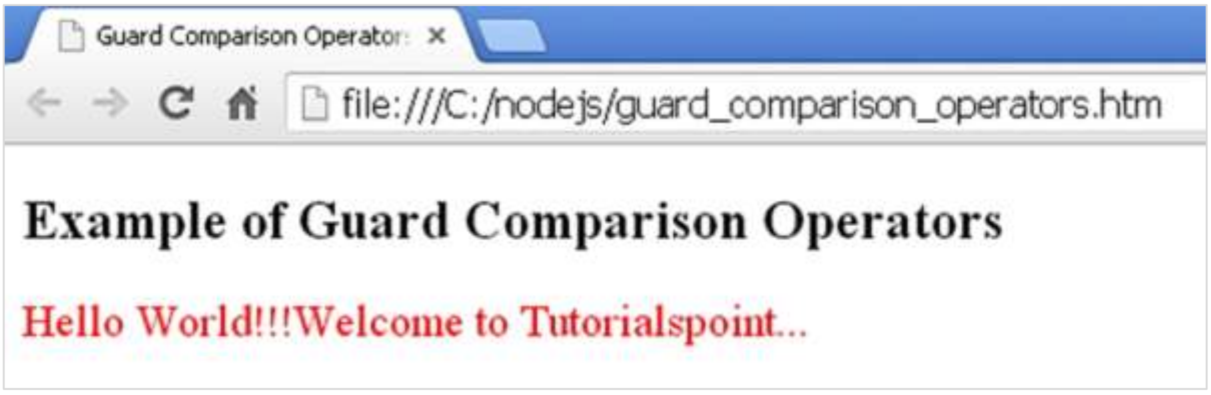
style.css

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

Output

Follow these steps to see how the above code works:

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


Guard Comparison Operator
file:///C:/nodejs/guard_comparison_operators.htm

Example of Guard Comparison Operators

Hello World!!!Welcome to Tutorialspoint...

LESS – Guard Logical Operators

Description

You can use the *and* keyword to work around logical operators with guards. You can combine the guard conditions using the *and* keyword and negate the conditions using the *not* keyword.

Example

The following example demonstrates the use of guard logical operators in the LESS file:

```
<!doctype html>
<head>
   <title>Guard Logical Operators</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Guard Logical Operators</h2>
   <p class="class1">Hello World...</p>
   <p class="class2">Welcome to Tutorialspoint...</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin (@a) when (@a > 50%) and (@a > 5px){
font-size: 14px;
}
.mixin (@a) when not (@a < 50%) and not (@a < 5px){
font-size: 20px;
}
.mixin (@a) {
  color: @a;
}

.class1 { .mixin(#FF0000) }
.class2 { .mixin(#555) }
```

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:

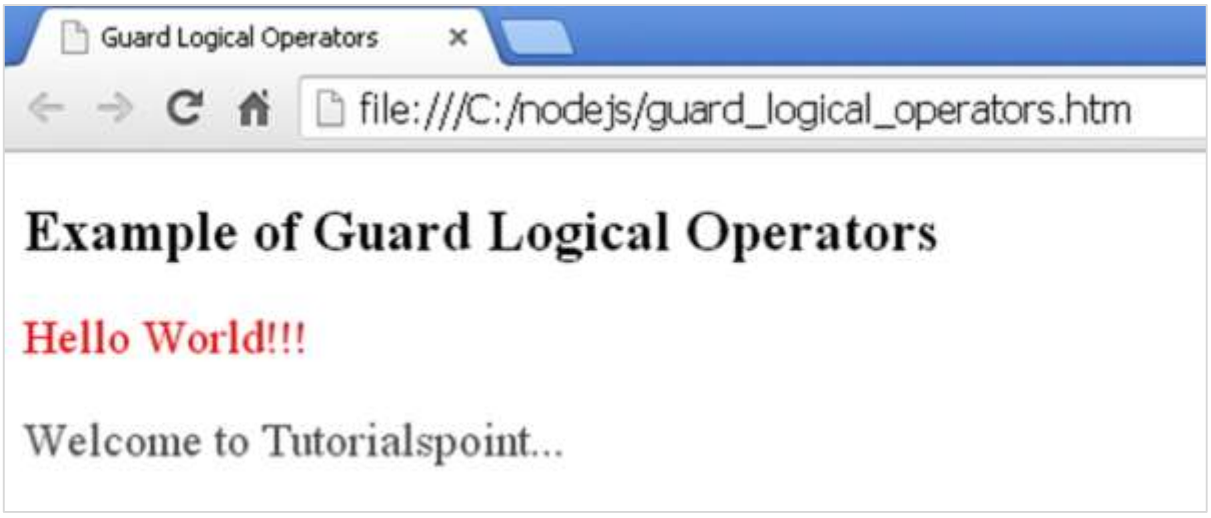
style.css

```
.class1 {
  font-size: 20px;
  color: #FF0000;
}
.class2 {
  font-size: 20px;
  color: #555;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Guard Logical Operators

Hello World!!!

Welcome to Tutorialspoint...

LESS – Type Checking Functions

Description

You can use type checking built-in functions to determine the value types for matching mixins. To do this, you can use the *is* functions. Following is the list of available functions:

- iscolor
- isnumber
- isstring
- iskeyword
- isurl

The functions listed above are into basic type checking. You can check whether a value is in a specific unit or not by using the following functions:

- ispixel
- ispercentage
- isem
- isunit

Example

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

```
<!doctype html>
<head>
   <title>Type Checking Functions</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Type Checking Functions</h2>
   <p class="myclass">Hello World!!!Welcome to Tutorialspoint...</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin (@a; @b: red) when (iscolor(@b)){
color:blue;
}
.mixin (@a) {
  font-size: @a;
}
.myclass { .mixin(20px) }
```

You can compile the *style.less* 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:

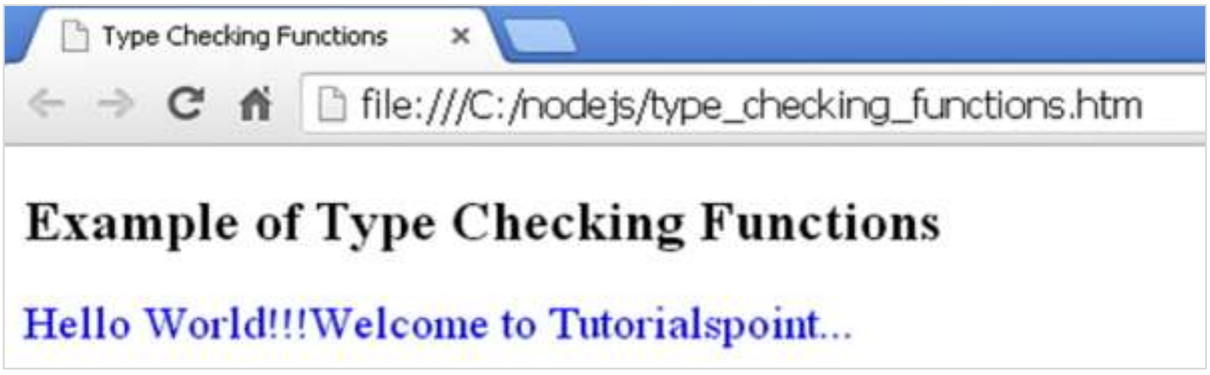
style.css

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

Output

Follow these steps to see how the above code works:

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



Type Checking Functions

file:///C:/nodejs/type_checking_functions.htm

Example of Type Checking Functions

Hello World!!!Welcome to Tutorialspoint...

LESS — Mixin Guards

Description

You can use the *default* function to match mixin with other mixing matches and create *conditional mixins* which look like *else* or *default* statements.

Example

The following example demonstrates the use of conditional mixins in the LESS file:

```
<!doctype html>
<head>
   <title>Conditional Mixins</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Conditional Mixins</h2>
   <p class="myclass">LESS enables customizable, manageable and reusable style
sheet for web site.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin (@a) when (@a > 22px){
color:blue;
}

.mixin (@a) when (@a <= 20px){
color:red;
}

.mixin (@a) {
  font-size: @a;
}

.myclass { .mixin(20px) }
```

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:

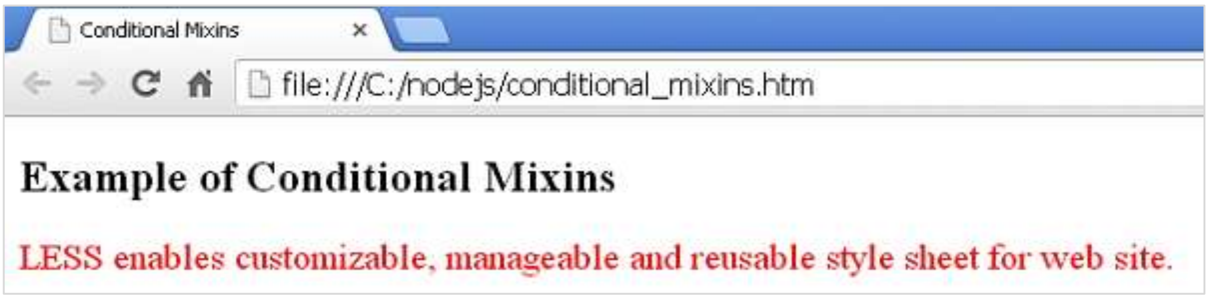
style.css

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

Output

Follow these steps to see how the above code works:

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



Example of Conditional Mixins

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

21. LESS — CSS Guards

Description

Guards are used to match simple values or a number of arguments on expressions. It is applied to the CSS selectors. It is syntax for declaring mixin and calling it immediately. To successfully bring out the **if** type statement; join this with feature **&**, which allows you to group multiple guards.

Example

The following example demonstrates the use of **css** guard in the LESS file:

css_guard.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="cont">
   <h2>Welcome to TutorialsPoint</h2>
</div>
<div class="style">
   <h3>The largest Tutorials Library on the web.</h3>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
@usedScope: global;
.mixin() {
  @usedScope: mixin;
  .cont when (@usedScope=global) {
    background-color: red;
    color: black;
  }
  .style when (@usedScope=mixin) {
    background-color: blue;
    color: white;
  }
  @usedScope: mixin;
}
.mixin();
```

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:

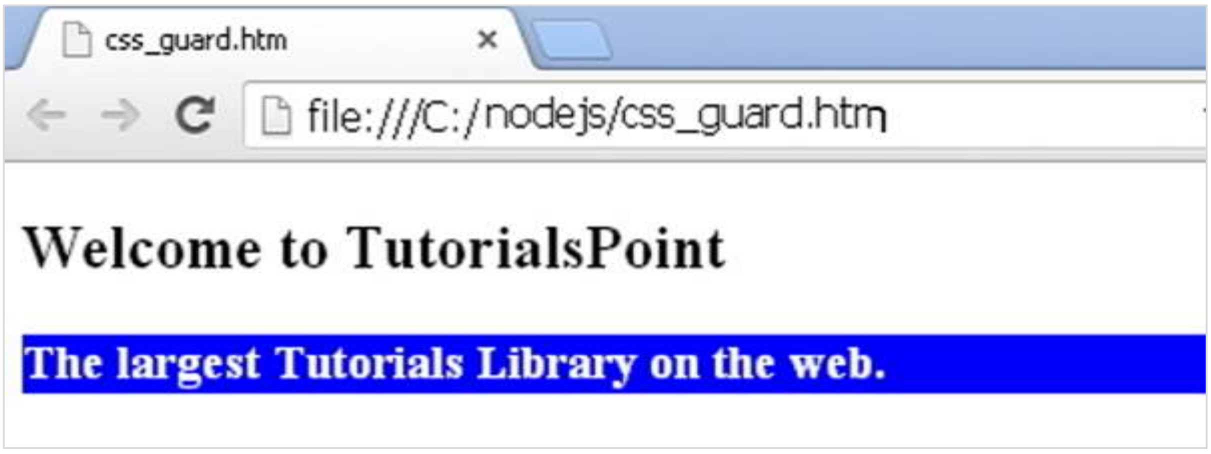
style.css

```
.style {
  background-color: blue;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

22. LESS — Loops

In this chapter, we will understand how Loops work in LESS. Loops statement allows us to execute a statement or group of statements multiple times. Various iterative/loop structures can be created when recursive mixin combine with **Guard Expressions** and **Pattern Matching**.

Example

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

loop_example.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="cont">
   <h2>Welcome to TutorialsPoint</h2>
   <p>The largest Tutorials Library on the web. </p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.cont(@count) when (@count > 0) {
  .cont((@count - 1));
  width: (25px * @count);
}
div {
  .cont(7);
}
```

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:

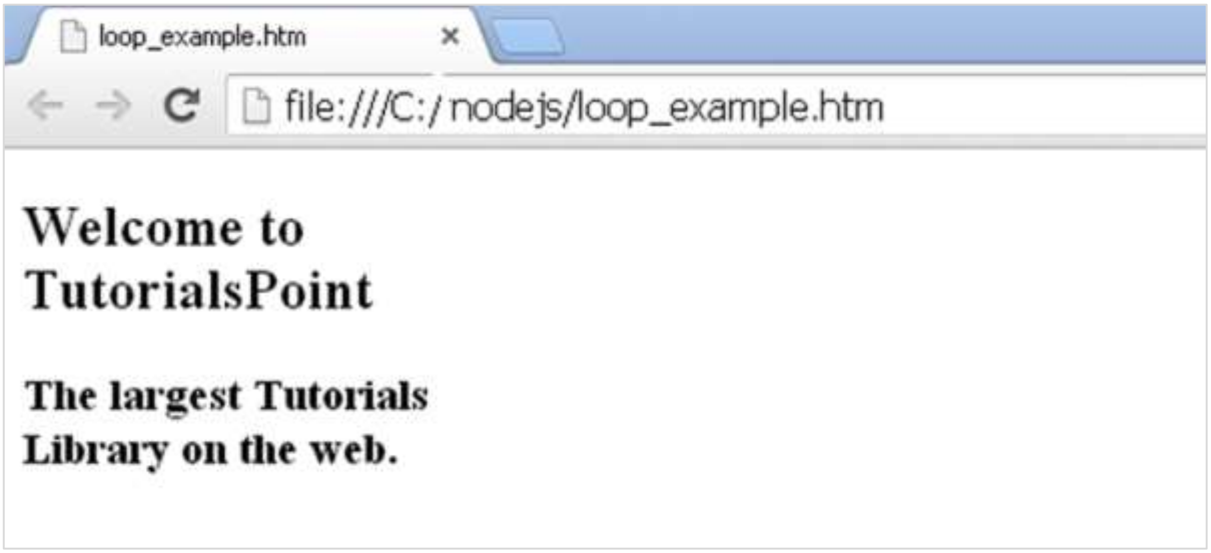
style.css

```
div {
  width: 25px;
  width: 50px;
  width: 75px;
  width: 100px;
  width: 125px;
  width: 150px;
  width: 175px;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

23. Less — Merge

Description

This feature in LESS allows the addition of values to comma or space separated list from multiple properties using a single property. It can be used for background and transform properties.

The following table describes the two types of functions supported by the Merge feature.

S.NO.	Types & Description
1	**Comma** It adds property value at the end.
2	**Space** It adds property value with space.

LESS — Merge Comma

Description

It adds property value to the very end.

Example

The following example demonstrates the use of merge comma feature in the LESS file:

```
<!doctype html>
<head>
   <title>Merge Comma</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Merge Comma</h2>
   <p class="class">Hello World!!!Welcome to Tutorialspoint...</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myfunc() {
  box-shadow+: 5px 5px 5px grey;
}
.class {
  .myfunc();
  box-shadow+: 0 0 5px #f78181;
}
```

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:

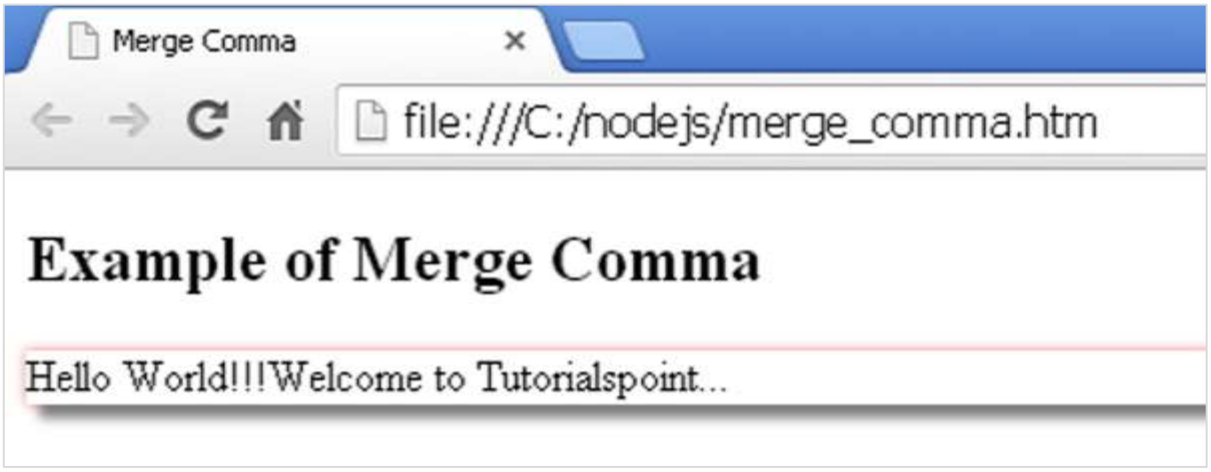
style.css

```
.class {
  box-shadow: 5px 5px 5px grey, 0 0 5px #f78181;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Merge Comma

Hello World!!!Welcome to Tutorialspoint...

LESS — Merge Space

Description

The Merge Space feature adds property value with space.

Example

The following example demonstrates the use of the merge space feature in the LESS file:

```
<!doctype html>
<head>
   <title>Merge Space</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example of Merge Space</h2>
   <p class="class">Hello World!!!Welcome to Tutorialspoint...</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.mixin() {
  transform+_: scale(1);
}
.class {
  .mixin();
  transform+_: rotate(2deg);
}
```

The merge uses **+** or **+_** flag to keep away from the unexpected joins on each join. The *transform* property modifies the space of CSS formatting model and can be used to rotate, scale, move, and perform other functions on elements.

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:

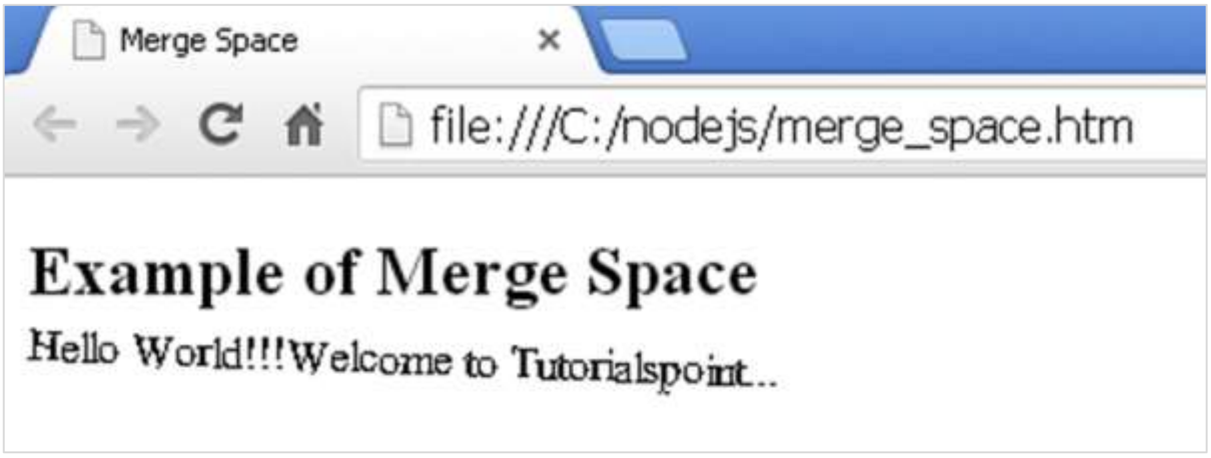
style.css

```
.class {
  transform: scale(1) rotate(2deg);
}
```

Output

Follow these steps to see how the above code works:

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



Example of Merge Space

Hello World!!!Welcome to Tutorialspoint..

24. LESS — Parent Selectors

In this chapter, let us understand how **Parent Selectors** work. It is possible to reference the parent selector by using the **&**(ampersand) operator. Parent selectors of a nested rule is represented by the **&** operator and is used when applying a modifying class or pseudo class to an existing selector.

The following table shows the types of parent selector:

S.NO.	Types & Description
1	**Multiple &** The **&** will represent the nearest selector and also all the parent selectors.
2	**Changing Selector Order** Prepending a selector to the inherited (parent) selectors is useful when selector ordering is changed.
3	**Combinatorial Explosion** The **&** can also produce all the possible permutation of selectors in a list separated by commas.

Example

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

```
<!doctype html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Parent Selector</title>
</head>
<body>
<h2>Welcome to TutorialsPoint</h2>
     <ul>
       <li><a>SASS</a></li>
       <li><a>LESS</a></li>
     </ul>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
a {
  color: #5882FA;
  &:hover {
    background-color: #A9F5F2;
  }
}
```

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

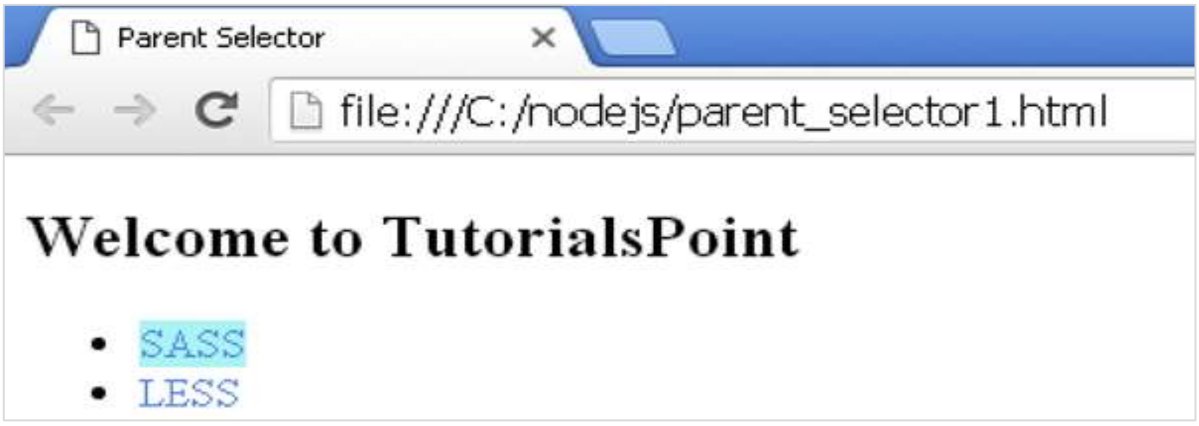
```
a {
  color: #5882FA;
}
a:hover {
  background-color: red;
}
```

In the above example, **&** refers to selector **a**.

Output

Follow these steps to see how the above code works:

- Save the above html code in **parent_selector1.htm** file.
- Open this HTML file in a browser, the following output gets displayed.



Welcome to TutorialsPoint

- SASS
- LESS

The *Parent selectors* operator has many uses like, when you need to combine the nested rule's selectors in other way than the default. Another typical use of **&** is to generate class names repeatedly. For more information click here.

LESS – Multiple &

Description

By using the **&** operator, it is possible to refer a parent selector repeatedly without using its name. Within a selector **&** can be used more than once.

Example

The following example demonstrates the use of multiple & in the LESS file:

```
<html>
<head>
    <title>Parent Selector</title>
     <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Welcome to TutorialsPoint</h2>
<p class="select">It is possible to reference the parent selector by using
&(ampersand) operator.</p>
<p class="select_class1">It is possible to reference the parent selector by using
&(ampersand) operator</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.select {
  & + & {
    color: #A9F5F2;
  }

  & & {
    color: #D0FA58;
  }
  && {
    color: #81BEF7;
  }

  &, &_class1 {
    color: #A4A4A4;
  }
}
```

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:

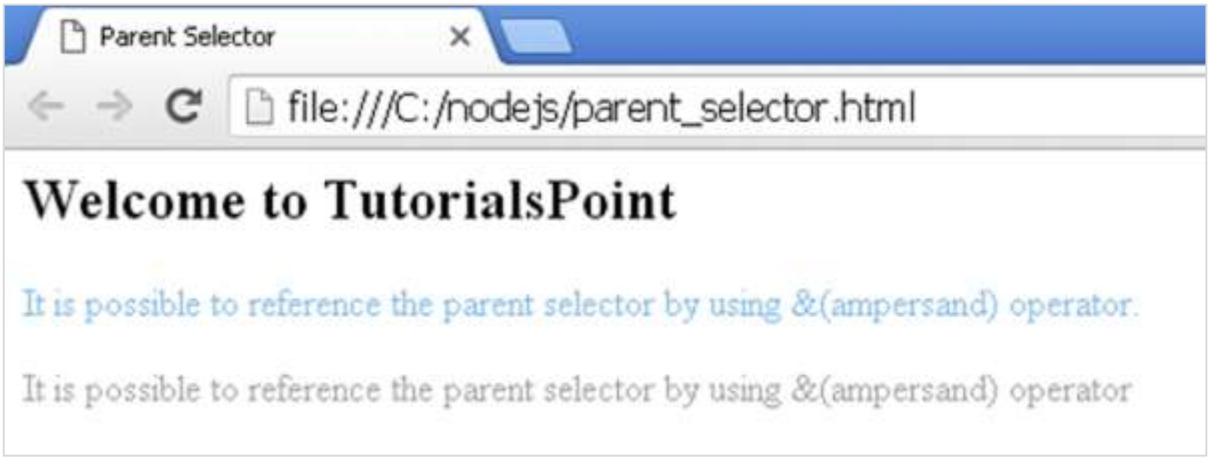
style.css

```
.select + .select {
  color: #A9F5F2;
}
.select .select {
  color: #D0FA58;
}
.select.select {
  color: #81BEF7;
}
.select,
.select_class1 {
  color: #A4A4A4;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

It is possible to reference the parent selector by using &(ampersand) operator.

It is possible to reference the parent selector by using &(ampersand) operator

The **&** will not just represent the nearest selector but represents all the parent selectors. For more information, click here.

LESS – Multiple &

Description

The **&** will not just represent the nearest selector but also represents all the parent selectors.

Example

The following example demonstrates the use to **&** representing all the parent selectors in the LESS file:

parent_selector.htm

```
<html>
<head>
     <link rel="stylesheet" href="multiple1.css" type="text/css" />
</head>
<body>
<div class="grand_parent">
<h3>Welcome to TutorialsPoint</h3>
<p class="parent"> It is possible to reference the parent selector by using
&(ampersand) operator.</p>
<p class="parent_class">It is possible to reference the parent selector by using
&(ampersand) operator.</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.grand_parent {
  .parent {
    & > & {
      color: #A9F5F2;
    }

    & & {
      color: #D0FA58;
    }

    && {
      color: #81BEF7;
    }

    &, &_class {
      color: #A4A4A4;
    }
  }
}
```

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:

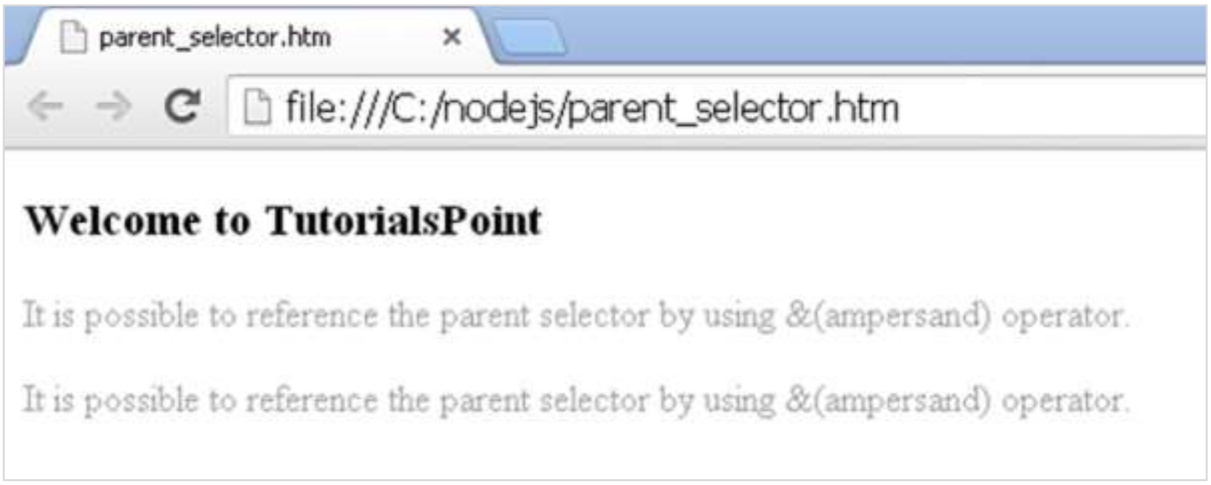
style.css

```
.grand_parent .parent > .grand_parent .parent {
  color: #A9F5F2;
}
.grand_parent .parent .grand_parent .parent {
  color: #D0FA58;
}
.grand_parent .parent.grand_parent .parent {
  color: #81BEF7;
}
.grand_parent .parent,
.grand_parent .parent_class {
  color: #A4A4A4;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

It is possible to reference the parent selector by using &(ampersand) operator.

It is possible to reference the parent selector by using &(ampersand) operator.

LESS — Changing Selector Order

Description

Prepending a selector to the inherited (parent) selectors is useful when selector ordering is changed. This is achieved by placing **&** after the selector. For instance, when you use Modernizer, you may wish to specify different rules depending on supported features.

Example

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

```
<html>
<head>
   <link rel="stylesheet" href="changing_selector_order.css" type="text/css" />
   <title>Parent Selector</title>
</head>
<body>
  <div class="header">
```

```
    <div class="menu">
      <h2>Welcome to TutorialsPoint</h2>
      <p>It is possible to reference the parent selector by using &(ampersand) operator.</p>
    </div>
  </div>
</body>
</html>
```

Now, create the *style.less* file.

style.less

```
.header {
  .menu {
    border-radius: 5px;
    border: 1px solid red;
    & {
      padding-left: 200px;
    }
  }
}
```

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

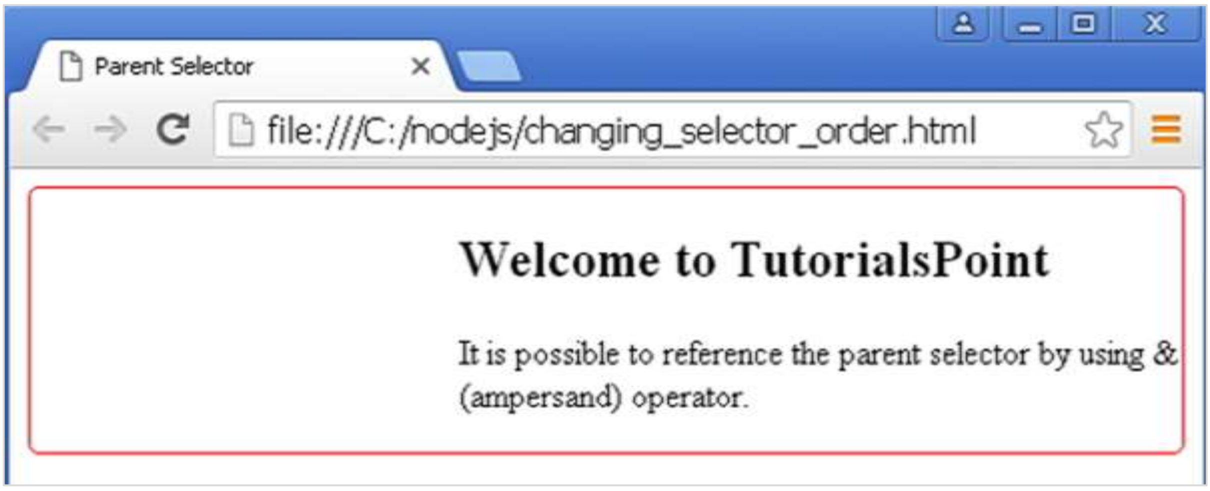
```
.header .menu {
  border-radius: 5px;
  border: 1px solid red;
  padding-left: 200px;
}
```

The **.no-borderradius &** selector will prepend **.no-borderradius** to its parent selector **.header .menu**

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

It is possible to reference the parent selector by using & (ampersand) operator.

LESS – Combinatorial Explosion

Description

The **&** can produce all the possible permutation of selectors in a list separated by commas.

Example

The following example demonstrates the use of **&** to produce all possible permutation of selectors in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <title>Combinatorial Explosion</title>
</head>
<body>
<p>This is first paragraph.</p>
<p>This is second paragraph which is adjecent to first paragraph ( i.e. p + p ).
This will be highlighted.</p>
<div>
This div is adjecent to second paragraph ( i.e. p + div ). This will be
highlighted.
</div>
```

```
<p>This is third paragraph adjecent to div ( i.e. p + div ). This will be
highlighted.</p>

<i>This is italic. This will not be highlighted since there is no (p + i) in
CSS</i>

<div>This is second div</div>

<div>This is div adjecent to second div ( i.e. div + div ). This will be
highlighted</div>

</body>

</html>
```

Next, create the *style.less* file.

style.less

```
p, div {
  color : red;
  font-family:Lucida Console;
  & + & {
    color : green;
    background-color: yellow;
    font-family: "Comic Sans MS";
  }
}
```

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:

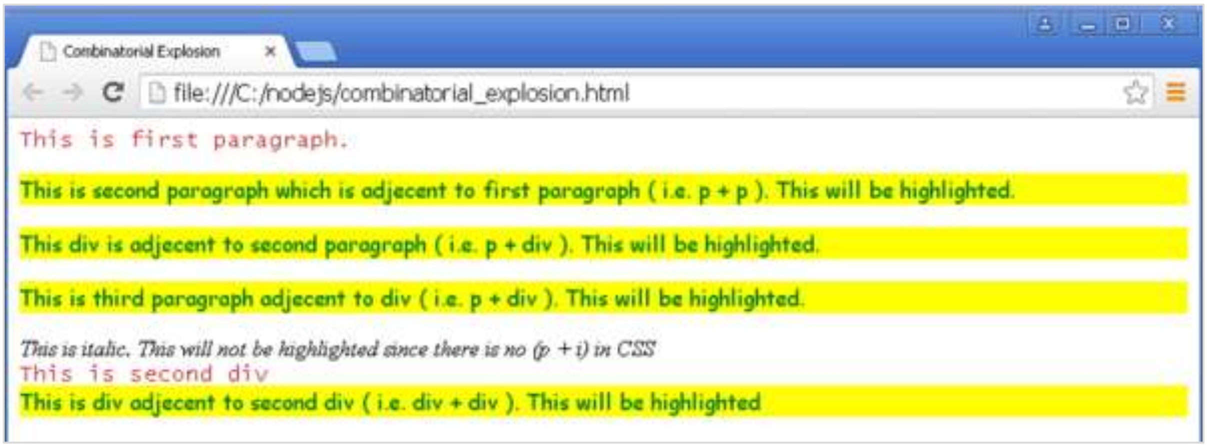
style.css

```
p,
div {
  color: red;
  font-family: Lucida Console;
}
p + p,
p + div,
div + p,
div + div {
  color: green;
  background-color: yellow;
  font-family: "Comic Sans MS";
}
```

Output

Follow these steps to see how the above code works:

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



```
This is first paragraph.
This is second paragraph which is adjecent to first paragraph ( i.e. p + p ). This will be highlighted.
This div is adjecent to second paragraph ( i.e. p + div ). This will be highlighted.
This is third paragraph adjecent to div ( i.e. p + div ). This will be highlighted.
This is italic. This will not be highlighted since there is no (p + i) in CSS
This is second div
This is div adjecent to second div ( i.e. div + div ). This will be highlighted
```

Functions

25. LESS — Misc Functions

Misc functions consist of a group of functions of a different kind. The following table lists all the types of misc functions:

S.NO.	Functions & Description
1	**color** It is a string which represents colors.
2	**image - size** It is used to examine the dimension of the image from the file.
3	**image - width** It examines the width of the image from the file.
4	**image - height** It examines the height of the image from the file.
5	**convert** A number is converted from one unit to another.
6	**data - uri** Data uri is uniform resource identifier (URI) schema which gets a resource inline on webpages.
7	**default** Default function returns true only when it is available inside the guard condition and does not match with any other mixin.
8	**unit** Default function returns true only when it is available inside the guard condition and does not match with any other mixin
9	**get - unit** get - unit function returns its unit where the argument is present with number and units.
10	**svg - gradient** svg-gradient is a transition of one color to another. It can add many colors to the same element.

LESS — Color Function

Description

It is a string which represents colors. Color can be set to background or foreground of the element. The color values are used to specify the color.

Example

The following example demonstrates the use of color function in the LESS file:

misc_example.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2 class="style">Welcome to TutorialsPoint</h2>
   <p class="style">The largest Tutorials Library on the web.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style{
color:(#fe2700)
};
```

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:

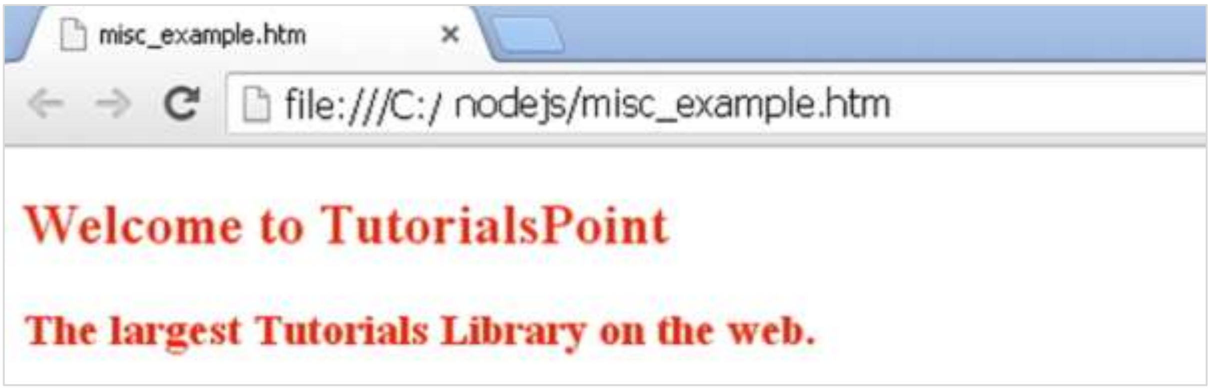
style.css

```
.style {
  color: #fe2700;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

Less — Image Size Function

Description

It is used to examine the dimension of the image from the file. It examines both the width and the height of the image.

Example

The following example demonstrates the use of image-size in the LESS file:

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
  <h2>Welcome to TutorialsPoint</h2>
  <h3>The largest Tutorials Library on the web.</h3>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
body{
   background-image:url("startup.jpg");
```

```
    image-size:image-size("startup.jpg");
}
```

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:

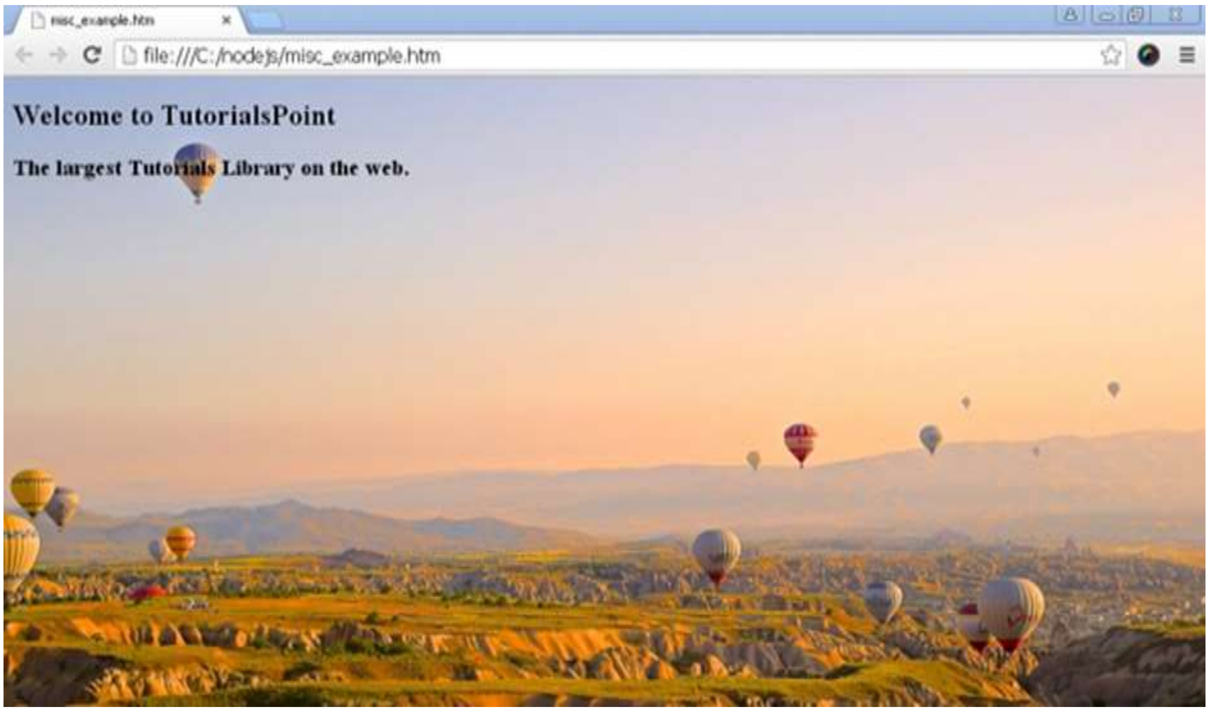
style.css

```
body {
  background-image: url("startup.jpg");
  image-size: 1200px 800px;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

LESS – Image Width Function

Description

It examines the width of the image from the file.

Example

The following example demonstrates the use of image width in the LESS file:

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
  <h2>Welcome to TutorialsPoint</h2>
  <h3>The largest Tutorials Library on the web.</h3>
</body>
</html>
```

Now, create the *style.less* file.

style.less

```
body{
  background-image:url("startup.jpg");
  width:image-width("startup.jpg");
}
```

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:

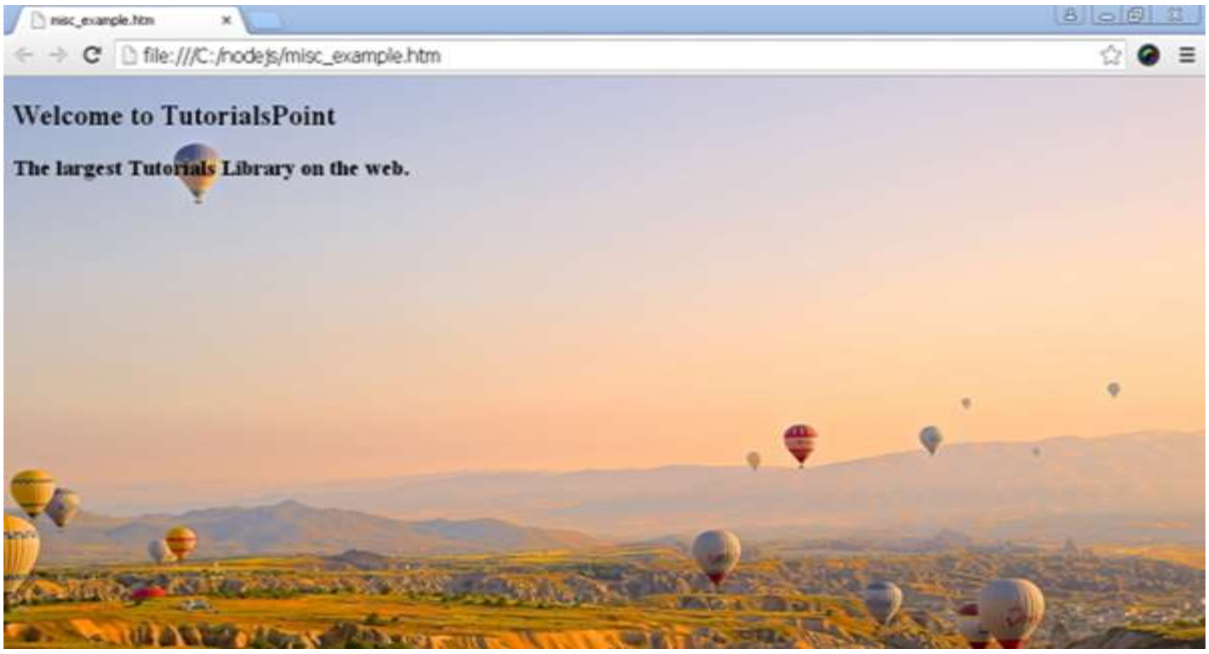
style.css

```
body {
  background-image: url("startup.jpg");
  width: 1200px;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

LESS – Image Height Function

Description

It examines the height of the image from the file.

Example

The following example demonstrates the use of image height in the LESS file:

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
  <h2>Welcome to TutorialsPoint</h2>
  <h3>The largest Tutorials Library on the web.</h3>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
body{
  background-image:url("startup.jpg");
  height:image-height("startup.jpg");
}
```

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:

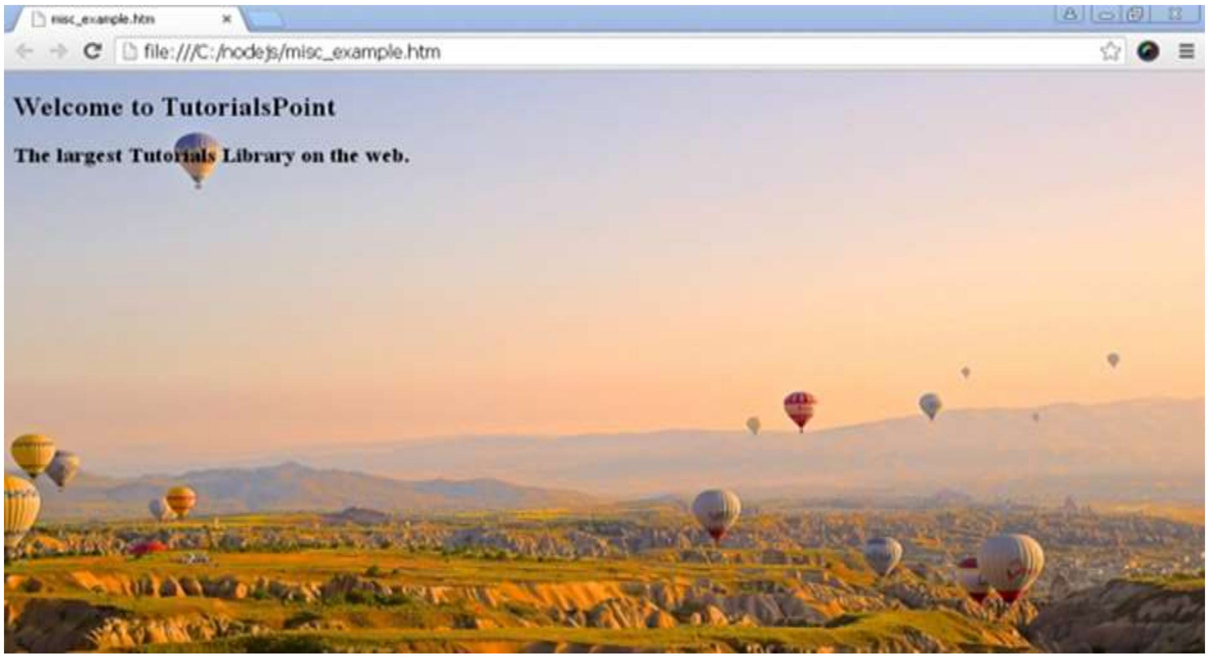
style.css

```
body {
  background-image: url("startup.jpg");
  height: 800px;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

LESS – Convert Function

Description

A number is converted from one unit to another. It consists two arguments; first argument passes number along with unit and second argument contain units. The number is converted when the unit is compatible. If the first argument is unchanged then the unit is not compatible.

Example

Below is the stylesheet file saved with extension *.less*; this is similar to a CSS file.

style.less

```
body{
  meter:convert(10cm, mm);
  time:convert(3s, "ms");
  no-unit:convert(5, mm);
}
```

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

```
body {
  meter: 100mm;
  time: 3000ms;
  no-unit: 5;
}
```

LESS – Data-uri Function

Description

Data uri is uniform resource identifier (URI) schema which inlines a resource in webpages. The node uses mime package to control the right MIME type when the MIME type is not given.

Parameter

- **url:** Inline the url of the file.
- **mimetype:** MIME type string.

Example

The following example demonstrates the use of data uri function in the LESS file:

misc_example.htm

```
<!doctype html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Welcome to TutorialsPoint</h2>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
body{
   background: data-uri('startup.jpg');
}
```

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:

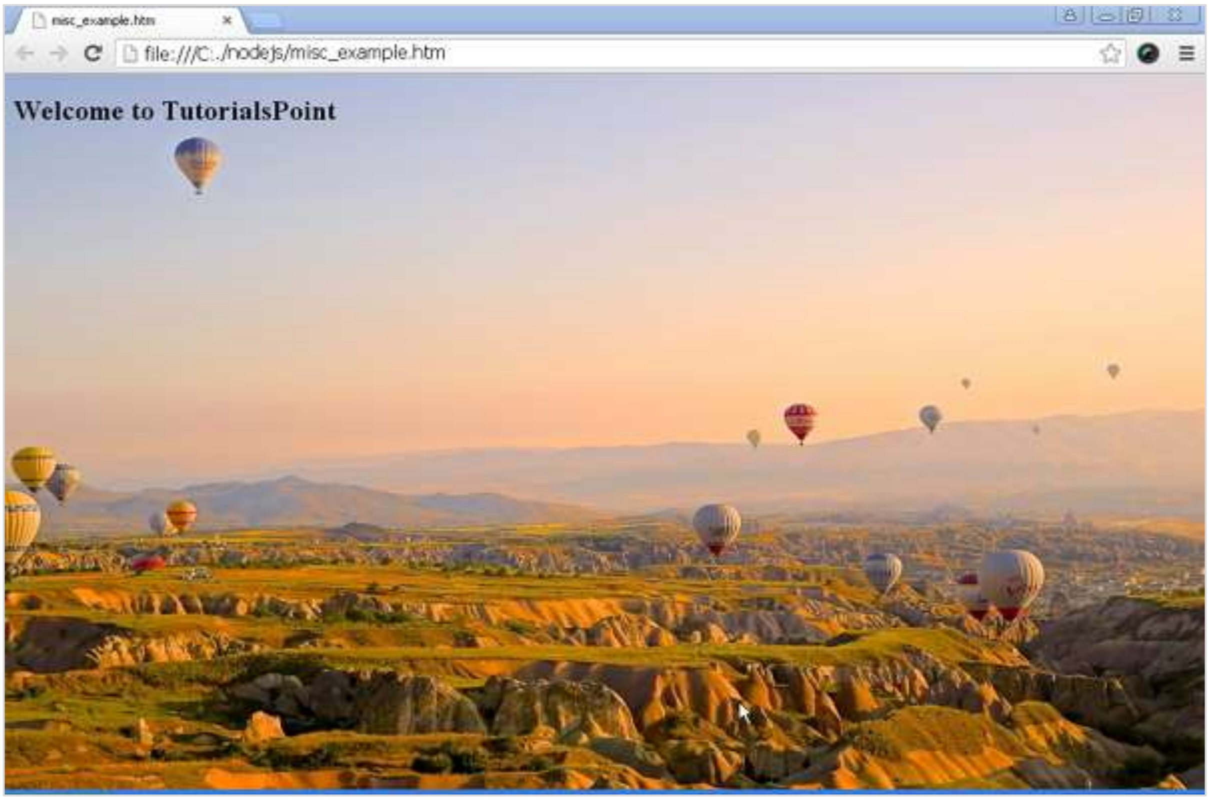
style.css

```
body {
  background: url('startup.jpg');
}
```

Output

Follow these steps to see how the above code works:

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


misc_example.htm
file:///C:./nodejs/misc_example.htm
Welcome to TutorialsPoint

LESS — Default Function

Description

The Default function returns true only when it is available inside the guard condition and does not match with any other mixin otherwise it returns false. It interprets as regular css when the default function is used outside the mixin guard condition.

Example

The following example demonstrates the use of default function in the LESS file:

Below is the stylesheet file saved with extension *.less*; this is similar to a CSS file.

style.less

```
.mixin(1)                   {a: 5}
.mixin(2)                   {b: 10}
.mixin(3)                   {c: 15}
```

```
.mixin(@a) when (default()) {d: @a}

div {
  .mixin(12);
}

div.style {
  .mixin(3);
```

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

```
div {
  d: 12;
}
div.special {
  c: 15;
}
```

LESS — Unit Function

Description

Less supports a number of measurements including absolute units such as inches, centimeters, points, and so on. It can change the unit of dimension.

Parameter

- **dimension:** Specifies a number with or without dimension.
- **unit:** It changes the unit.

Example

The following example describes the use of color function in the LESS file:

misc_example.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="style">
   <h2>Welcome to TutorialsPoint</h2>
   <p>The largest Tutorials Library on the web.</p>
 </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.style{
  font-size:unit(30, px);
}
```

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:

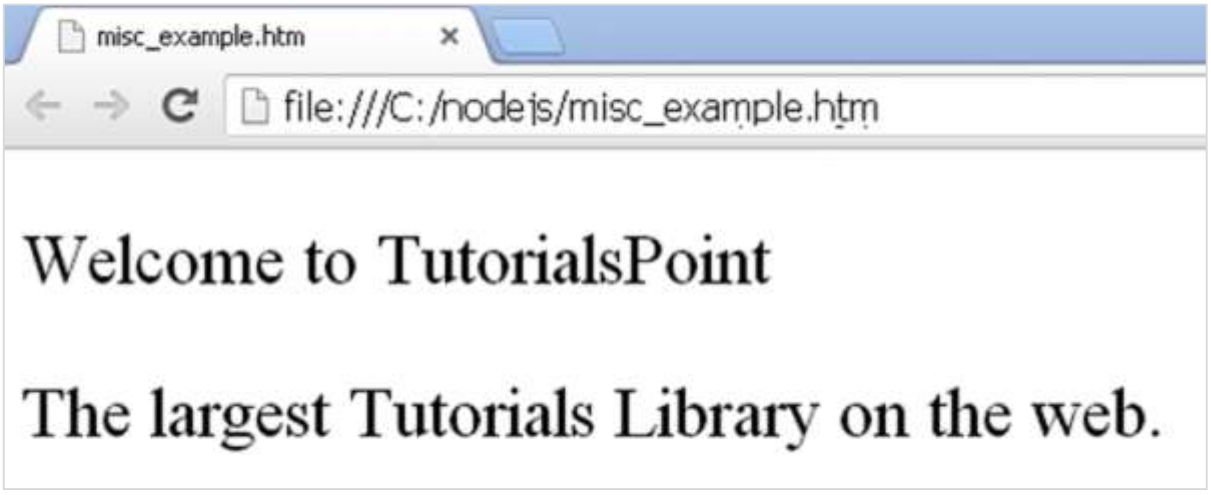
style.css

```
.style{
  font-size: 30px;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

Less — get - unit Function

Description

The *get - unit* function returns its unit where the argument is present with number and units. If the argument does not contain any unit then it returns empty value.

Example

The following example demonstrates the use of get unit in the LESS file:

misc_example.htm

```
<!doctype html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="list">
   <h2>Welcome to TutorialsPoint</h2>
   <p>The largest Tutorials Library on the web.</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.list{
  font-size:get-unit(30px);
  padding-left:get-unit(25);
}
```

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:

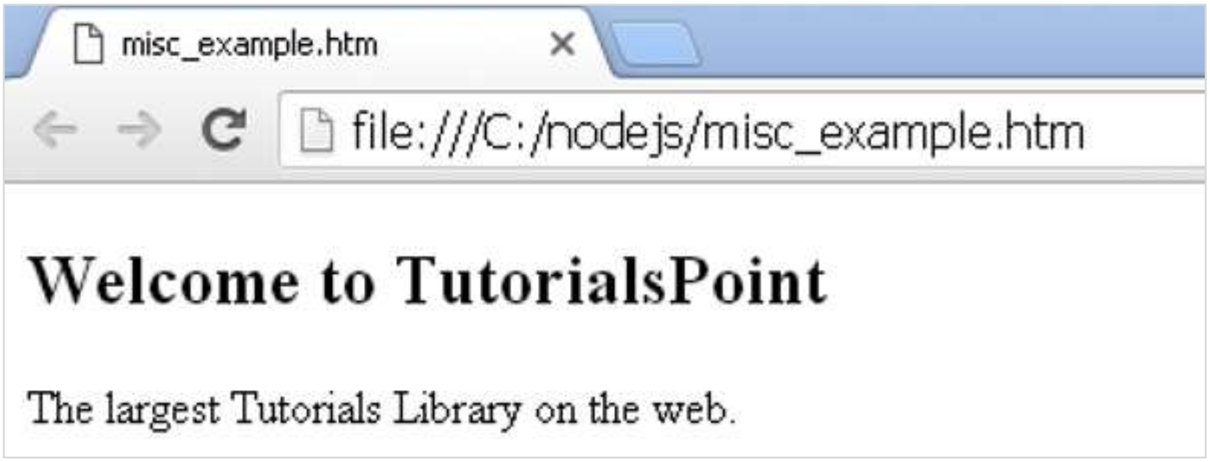
style.css

```
.list {
  font-size: px;
  padding-left: ;
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

Less — svg gradient Function

Description

The **svg-gradient** is a transition of one color to another. It can add many colors to the same element. It consists at least three parameters:

- first parameter identifies the gradient type and direction.
- other parameters list its position and color.

The colors specified at first and last position are optional. The direction can be set — to bottom, to right, to bottom right, to top right, ellipse or ellipse at center.

Parameters - colors stops in list:

- **list:** lists all colors and their position.
- **escaped value or list of identifiers:** sets the direction.

Parameters - color stops in arguments:

- **escaped value or list of identifiers:** sets the direction.
- **color[percentage] pair:** first color and its respective position.
- **color percent pair:** second color and its respective position.

Example

The following example demonstrates the use of **svg** gradient in the LESS file:

misc_example.htm

```
<!doctype html>
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div class="style">
   <h2>Welcome to TutorialsPoint</h2>
   <p>The largest Tutorials Library on the web.</p>
</div>
</body>
</html>
```

Create the *style.less* file.

style.less

```
.style {
  @style: orange, green 30%, #DAA520;
  background-image: svg-gradient(ellipse, @style);
}
```

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:

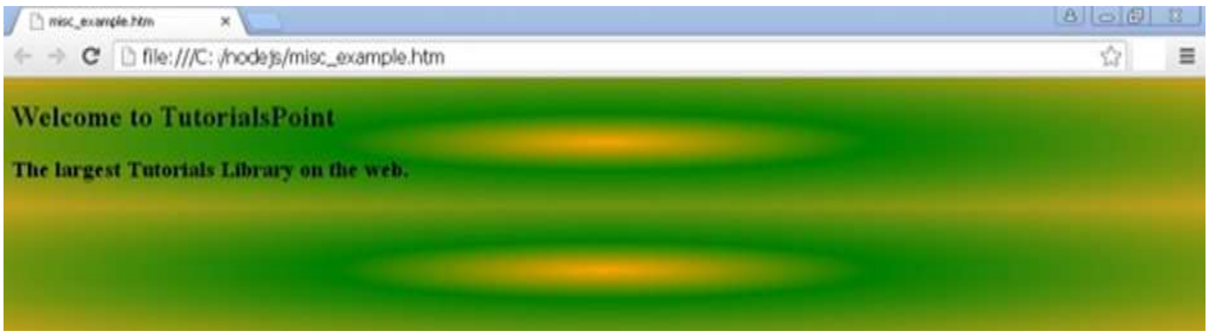
style.css

```
.style {
  background-image: 
url('data:image/svg+xml,%3C%3Fxml%20version%3D%221.0%22%20%3F%3E%3Csvg%20xmlns%
3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20version%3D%221.1%22%20width%3D%2
2100%25%22%20height%3D%22100%25%22%20viewBox%3D%220%200%201%201%22%20preserveAs
pectRatio%3D%22none%22%3E%3CradialGradient%20id%3D%22gradient%22%20gradientUnit
s%3D%22userSpaceOnUse%22%20cx%3D%2250%25%22%20cy%3D%2250%25%22%20r%3D%2275%25%2
2%3E%3Cstop%20offset%3D%220%25%22%20stop-
color%3D%22%23ffa500%22%2F%3E%3Cstop%20offset%3D%2230%25%22%20stop-
color%3D%22%23008000%22%2F%3E%3Cstop%20offset%3D%22100%25%22%20stop-
color%3D%22%23daa520%22%2F%3E%3C%2FradialGradient%3E%3Crect%20x%3D%22-
50%22%20y%3D%22-
50%22%20width%3D%22101%22%20height%3D%22101%22%20fill%3D%22url(%23gradient)%22%
20%2F%3E%3C%2Fsvg%3E');
}
```

Output

Follow these steps to see how the above code works:

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



Welcome to TutorialsPoint

The largest Tutorials Library on the web.

26. Less — String Functions

Description

Less supports some of the string functions as listed below:

- escape
- e
- % format
- replace

The following table describes the above string functions along with description.

S.NO.	Types & Description	Example

1	**Escape**
	It encodes a string or information by using URL encoding on special characters. You could not encode some characters such as **,** , **/** , **?** , **@** , **&** , **+** , **~** , **!** , **$** , **'** and some characters you can encode such as \ , **#** , **^** , **(** , **)** , **{** , **}** , **:** , **>** , **<** , **]** , **[** and **=**.
		`escape("Hello!! welcome to Tutorialspoint!")`
		
		It outputs escaped string as:
		
		`Hello%21%21%20welcome%20to%20Tutorialspoint%21`

2	**e**
	It is a string function which uses string as parameter and returns the information without quotes. It is a CSS escaping which uses ~*"some content"* escaped values and numbers as parameters.
		`filter: e("Hello!! welcome to Tutorialspoint!");`
		
		It outputs escaped string as:
		
		`filter: Hello!! welcome to Tutorialspoint!;`

3	**% format** This function formats a string. It can be written with the following format:	`format-a-d: %("myvalues: %a myfile: %d", 2 + 3, "mydir/less_demo.less");`

	%(string, arguments ...)	It outputs the formatted string as: `format-a-d: "myvalues: 5 myfile: "mydir/less_demo.less"";`

27. LESS — List Functions

Description

LESS consists of list functions which are used to specify length of the list and position of the value in the list.

Following table lists the List functions used in LESS:

S.NO.	Function & Description
1	**Length** It will take a comma or space separated list of values as parameter.
2	**Extract** It will return the value at a specified position in a list.

LESS — List Length Function

Description

It will take a comma or space separated list of values as parameter and returns an integer representing the number of elements in a list.

Example

The following example demonstrates the use of length function in the LESS file:

```
<!doctype html>
<head>
   <title>Less Length Function</title>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
   <h2>Example using Length Function</h2>
   <p>LESS enables customizable, manageable and reusable style sheet for web site.</p>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
p{
@list: "audi", "benz", "toyota","honda", "mahindra", "suzuki","volkswagen",
"renault","bmw","tata","ford","skoda";
@val: length(@list);
font-size:@val;
}
```

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: 12;
}
```

Output

Follow these steps to see how the above code works:

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

LESS — List Extract Function

Description

It will return the value at a specified position in a list. It takes **list** (list of values separated by a comma or space) and **index** (integer specifying a position of an element to return) as parameters.

Example

The following example demonstrates the use of extract function in the LESS file:

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

Next, create the *style.less* file.

style.less

```
p{
@list: 10px, 20px, 30px, 40px;
@val: extract(@list, 2);
font-size:@val;
color:#F79F81;
}
```

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: 20px;
  color: #F79F81;
}
```

Output

Follow these steps to see how the above code works:

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

28. Less — Math Functions

Description

Math functions includes methods which are used for performing numeric operations such as round, square root, power value, modulus, percentage, etc.

Following table shows Math Functions used in LESS:

S.NO.	Function & Description	Example

1	**ceil**
	It rounds up the number to next highest integer.
		`ceil(0.7)`
		it rounds the number to:
		`1`

2	**floor**
	It rounds down the number to next lowest integer.
		`floor(3.3)`
		it rounds the number to:
		`3`

3	**percentage**
	It transforms the floating point number to percentage string.
		`percentage(0.2)`
		it converts the number to percentage string as:
		`20%`

4	**round**
	It rounds a floating point number.
		`round(3.77)`
		it converts the number to the rounding value as:
		`4`

5	**sqrt**
	It returns the square root of a number.
		`sqrt(25)`
		it defines the square root of a number as:
		`5`

6	**abs**
	It provides the absolute value of a number.
		`abs(30ft)`
		it displays the absolute value as:
		`30ft`

7	**sin**
	It returns radians on numbers.
		`sin(2)`
		it calculates the sine value as:
		`0.90929742682`

8	**asin**
	It specifies arcsine (inverse of sine) of a number which returns value between -pi/2 and pi/2.
		`asin(1)`
		it calculates the asin value as:
		`1.5707963267948966`

9	**cos**
	It returns cosine of the specified value and determines radians on numbers without units.
		`cos(2)`
		it calculates the cos value as:
		`-0.4161468365471424`

10	**acos** It specifies arccosine (inverse of	`acos(1)`

	cosine) of a number which returns value between 0 and pi.	it calculates the acos value as: `0`
11	**tan** It specifies tangent of the number.	`tan(60)` it calculates the tan value as: `0.320040389379563`
12	**atan** It specifies arctangent (inverse of tangent) of a specified number.	`atan(1)` it displays atan value as: `0.7853981633974483`
13	**pi** It returns the pi value.	`pi()` it determines the pi value as: `3.141592653589793`
14	**pow** It specifies the value of first argument raised to the power of second argument.	`pow(3,3)` it specifies the power value as: `27`
15	**mod** It returns modulus of first argument with respect to the second argument. It also handles negative and floating point numbers.	`mod(7,3)` it returns the modulus value as: `1`

16	**min**
	It specifies the smallest value of one or more arguments.
		`min(70,30,45,20)`
		it returns the minimum value as:
		`20`

17	**max**
	It specifies the highest value of one or more arguments.
		`max(70,30,45,20)`
		it returns the maximum value as:
		`70`

29. Less – Type Functions

In this chapter, we will understand the importance of **Type Functions** in LESS. They are used to determine the type of the value.

The following table shows the *Type Functions* used in LESS.

S.N.	Type Functions & Description	Example

1	**isnumber**
	It takes a value as parameter and returns *true*, if it's a number or *false* otherwise.
		`isnumber(1234);       // true`
		`isnumber(24px);       // true`
		`isnumber(7.8%);       // true`
		`isnumber(#fff);       // false`
		`isnumber(red);        // false`
		`isnumber("variable"); // false`
		`isnumber(keyword);    // false`
		`isnumber(url(...));   // false`

2	**isstring**
	It takes a value as parameter and returns *true*, if it's a string or *false* otherwise.
		`isstring("variable"); // true`
		`isstring(1234);       // false`
		`isstring(24px);       // false`
		`isstring(7.8%);       // false`
		`isstring(#fff);       // false`
		`isstring(red);        // false`
		`isstring(keyword);    // false`
		`isstring(url(...));   // false`

3	**iscolor**
	It takes a value as parameter and returns *true*, if value is a color or *false* if it's not.
		`iscolor(#fff);        // true`
		`iscolor(red);         // true`
		`iscolor(1234);        // false`
		`iscolor(24px);        // false`
		`iscolor(7.8%);        // false`
		`iscolor("variable");  // false`
		`iscolor(keyword);     // false`
		`iscolor(url(...));    // false`

4	**iskeyword**
	It takes a value as parameter and returns *true*, if value is a keyword or *false* if it's not.
		`iskeyword(keyword);   // true`
		`iskeyword(1234);      // false`
		`iskeyword(24px);      // false`
		`iskeyword(7.8%);      // false`
		`iskeyword(#fff);      // false`
		`iskeyword(red) ;      // false`
		`iskeyword("variable");// false`
		`iskeyword(url(...));  // false`

5	**isurl**
	It takes a value as parameter and returns *true*, if value is a url or *false* if it's not.
		`isurl(url(...));      // true`
		`isurl(keyword);       // false`
		`isurl(1234);          // false`
		`isurl(24px);          // false`
		`isurl(7.8%);          // false`
		`isurl(#fff);          // false`
		`isurl(red) ;          // false`
		`isurl("variable");    // false`

6	**ispixel**
	It takes a value as parameter and returns *true*, if value is a number in pixels or *false* otherwise.
		`ispixel(24px);          // true`
		`ispixel(1234);          // false`
		`ispixel(7.8%);          // false`
		`ispixel(keyword);       // false`
		`ispixel(#fff);          // false`
		`ispixel(red) ;          // false`
		`ispixel("variable");    // false`
		`ispixel(url(...));      // false`

7	**isem**
	It takes a value as parameter and returns *true*, if value is an em value or *false* if it's not.
		`isem(0.5em);            // true`
		`isem(1234);             // false`
		`isem(24px);             // false`
		`isem(keyword);          // false`
		`isem(#fff);             // false`
		`isem(red) ;             // false`
		`isem("variable");       // false`
		`isem(url(...));         // false`

8	**ispercentage**
	It takes a value as parameter and returns *true*, if value is in percentage or returns *false*, if value is not in percentage.
		`ispercentage(7.5%);        // true`
		`ispercentage(url(...));    // false`
		`ispercentage(keyword);     // false`
		`ispercentage(1234);        // false`
		`ispercentage(24px);        // false`
		`ispercentage(#fff);        // false`
		`ispercentage(red) ;        // false`
		`ispercentage("variable"); // false`

9	**isunit**
	It returns *true* if a value is a number in specified units provided as parameter or it will return *false* if value is not a number in specified units.
		`isunit(10px, px);       // true`
		`isunit(5rem, rem);      // true`
		`isunit(7.8%, '%');      // true`
		`isunit(2.2%, px);       // false`
		`isunit(24px, rem);      // false`
		`isunit(48px, "%");      // false`
		`isunit(1234, em);       // false`
		`isunit(#fff, pt);       // false`
		`isunit("mm", mm);       // false`

10	**isruleset**
	It takes a value as parameter and returns *true*, if value is a ruleset or *false* otherwise.
		`@rules: {`
		`    color: green;`
		`}`
		`isruleset(@rules);      // true`
		`isruleset(1234);        // false`
		`isruleset(24px);        // false`
		`isruleset(7.8%);        // false`
		`isruleset(#fff);        // false`
		`isruleset(blue);        // false`
		`isruleset("variable");  // false`
		`isruleset(keyword);     // false`
		`isruleset(url(...));    // false`

30. Less — Color Defination Functions

Description

LESS provides number of useful color functions to alter and manipulate colors in different ways. LESS supports some of the Color Definition Functions as shown in the table below:

S.NO.	Function & Description	Example

1	**rgb**
	It creates color from red, green and blue values. It has following parameters:
	• **red**: It contains integer between 0 - 255 or percentage between 0 - 100%.
	• **green**: It contains integer between 0 - 255 or percentage between 0 - 100%.
	• **blue**: It contains integer between 0 - 255 or percentage between 0 - 100%.
		`rgb(220,20,60)`
		it converts color with rgb values as:
		`#dc143c`

2	**rgba**
	It determines color from red, green, blue and alpha values. It has the following parameters:
	• **red**: It contains integer between 0 - 255 or percentage between 0 - 100%.
	• **green**: It contains integer between 0 - 255 or percentage between 0 - 100%.
	• **blue**: It contains integer between 0 - 255 or percentage between 0 - 100%.
		`rgba(220,20,60, 0.5)`
		it converts color object with rgba values as:
		`rgba(220, 20, 60, 0.5)`

	• **alpha**: It contains number between 0 - 1 or percentage between 0 - 100%.	

3	**argb**
	It defines hex representation of color in **#AARRGGBB** format. It uses the following parameter:
	• **color**: It specifies color object.
		`argb(rgba(176,23,31,0.5))`
		it returns the argb color as:
		`#80b0171f`

4	**hsl**
	It generates the color from hue, saturation and lightness values. It has following parameters:
	• **hue**: It contains integer between 0 - 360 which represents degrees.
	• **saturation**: It contains number between 0 - 1 or percentage between 0 - 100%.
	• **lightness**: It contains number between 0 - 1 or percentage between 0 - 100%.
		`hsl(120,100%, 50%)`
		it returns the color object using HSL values as:
		`#00ff00`

5	**hsla**
	It generates the color from hue, saturation, lightness and alpha values. It has the following parameters:
	• **hue**: It contains integer between 0 - 360 which represents degrees.
	• **saturation**: It contains number between 0 - 1 or percentage between 0 - 100%.
		`hsla(0,100%,50%,0.5)`
		it specifies the color object using HSLA values as:
		`rgba(255, 0, 0, 0.5);`

	• **lightness**: It contains number between 0 - 1 or percentage between 0 - 100%. • **alpha**: It contains number between 0 - 1 or percentage between 0 - 100%.	

6	**hsv**
	It produces the color from hue, saturation and value values. It contains following parameters:
	
	• **hue**: It contains integer between 0 - 360 which represents degrees.
	
	• **saturation**: It contains number between 0 - 1 or percentage between 0 - 100%.
	
	• **value**: It contains number between 0 - 1 or percentage between 0 - 100%.
		`hsv(80,90%,70%)`
		
		it converts color object with hsv values as:
		
		`#7db312`

7	**hsva** It produces the color from hue, saturation, value and alpha values. It uses the following parameters:
	
	• **hue**: It contains integer between 0 - 360 which represents degrees.
	
	• **saturation**: It contains number between 0 - 1 or percentage between 0 - 100%.
	
	• **value**: It contains number between 0 - 1 or percentage between 0 - 100%.
	
	• **alpha**: It contains number between 0 - 1 or percentage between 0 - 100%.
		`hsva(80,90%,70%,0.6)`
		
		it specifies color object with hsva values as:
		
		`rgba(125, 179, 18, 0.6)`

31. Less — Color Channel Functions

In this chapter, we will understand the importance of Color Channel Functions in LESS. LESS supports few in-built functions which set value about a channel. The channel set the value depending on the color definition. The HSL colors have hue, saturation and lightness channel and the RGB color have red, green and blue channel. The following table lists out all the color channel functions:

S.NO.	Function & Description	Example

1	**hue**
	In HSL color space, the hue channel is extracted off the color object.
		`background: hue(hsl(75, 90%, 30%));`
		It compiles in the CSS as shown below :
		`background: 75;`

2	**saturation**
	In HSL color space, the saturation channel is extracted off the color object.
		`background: saturation(hsl(75, 90%, 30%));`
		It compiles in the CSS as shown below:
		`background: 90%;`

3	**lightness**
	In HSL color space, the lightness channel is extracted off the color object.
		`background: lightness(hsl(75, 90%, 30%));`
		It compiles in the CSS as shown below:
		`background: 30%;`

4	**hsvhue** In HSV color space, the hue channel is extracted off the color object.	`background: hsvhue(hsv(75, 90%, 30%));`

		It compiles in the CSS as shown below: `background: 75;`
5	**hsvsaturation** In HSL color space, the saturation channel is extracted off the color object.	`background: hsvsaturation(hsv(75, 90%, 30%));` It compiles in the CSS as shown below: `background: 90%;`
6	**hsvvalue** In HSL color space, the value channel is extracted off the color object.	`background: hsvvalue(hsv(75, 90%, 30%));` It compiles in the CSS as shown below: `background: 30%;`
7	**red** The red channel is extracted off the color object.	`background: red(rgb(5, 15, 25));` It compiles in the CSS as shown below: `background: 5;`
8	**green** The green channel is extracted off the color object.	`background: green(rgb(5, 15, 25));` It compiles in the CSS as shown below: `background: 15;`

9	**blue**
	The blue channel is extracted off the color object.
		`background: blue(rgb(5, 15, 25));`
		It compiles in the CSS as shown below:
		`background: 25;`

10	**alpha**
	The alpha channel is extracted off the color object.
		`background: alpha(rgba(5, 15, 25, 1.5));`
		It compiles in the CSS as shown below:
		`background: 2;`

11	**luma**
	luma value is calculated off a color object.
		`background: luma(rgb(50, 250, 150));`
		It compiles in the CSS as shown below:
		`background: 71.2513323%;`

12	**luminance**
	The luma value is calculated without gamma correction.
		`background: luminance(rgb(50, 250, 150));`
		It compiles in the CSS as shown below:
		`background: 78.53333333%;`

32. Less – Color Operation

Description

LESS provides number of useful operation functions to alter and manipulate colors in different ways and take parameters in the same units. LESS supports some of the Color Operation Functions as shown in the table below:

S.NO.	Directives & Description
1	**saturate** It varies the intensity or saturation of a color in the element.
2	**desaturate** It decreases the intensity or saturation of a color in the element.
3	**lighten** It increases the lightness of a color in the element.
4	**darken** It varies the intensity or saturation of a color in the element.
5	**fadein** It increase the opacity for selected elements.
6	**fadeout** It decreases the opacity for selected elements.
7	**fade** It is used to set the transparency of a color for selected elements.
8	**spin** It is used to rotate the angle of a color for selected elements.
9	**mix** It mixes the two colors along with the opacity.

10	**tint** It mixes the color with white as you decrease the proportion of the color.
11	**shade** It mixes the color with black as you decrease the proportion of the color.
12	**greyscale** It discards the saturation from a color in the selected elements.
13	**contrast** It sets the contrast for the colors in the element.

Less — Saturate

Description

It varies the intensity or saturation of a color in the element. It has the following parameters:

- **color**: It represents color object.
- **amount**: It contains percentage between 0 - 100%.
- **method**: It is optional parameter which is used for adjustment to be relative to current value by setting it to *relative*.

Example

The following example demonstrates the use of saturate color operation in the LESS file:

```
<html>
<head>
   <title>Saturate</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Saturate Color Operation</h2>
   <div class="myclass1">
   <p>color :<br> #426105</p>
   </div><br>
   <div class="myclass2">
      <p>result :<br> #446600</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsl(80, 90%, 20%);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: saturate(hsl(80, 90%, 20%), 10%);
   color:white;
}
```

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:

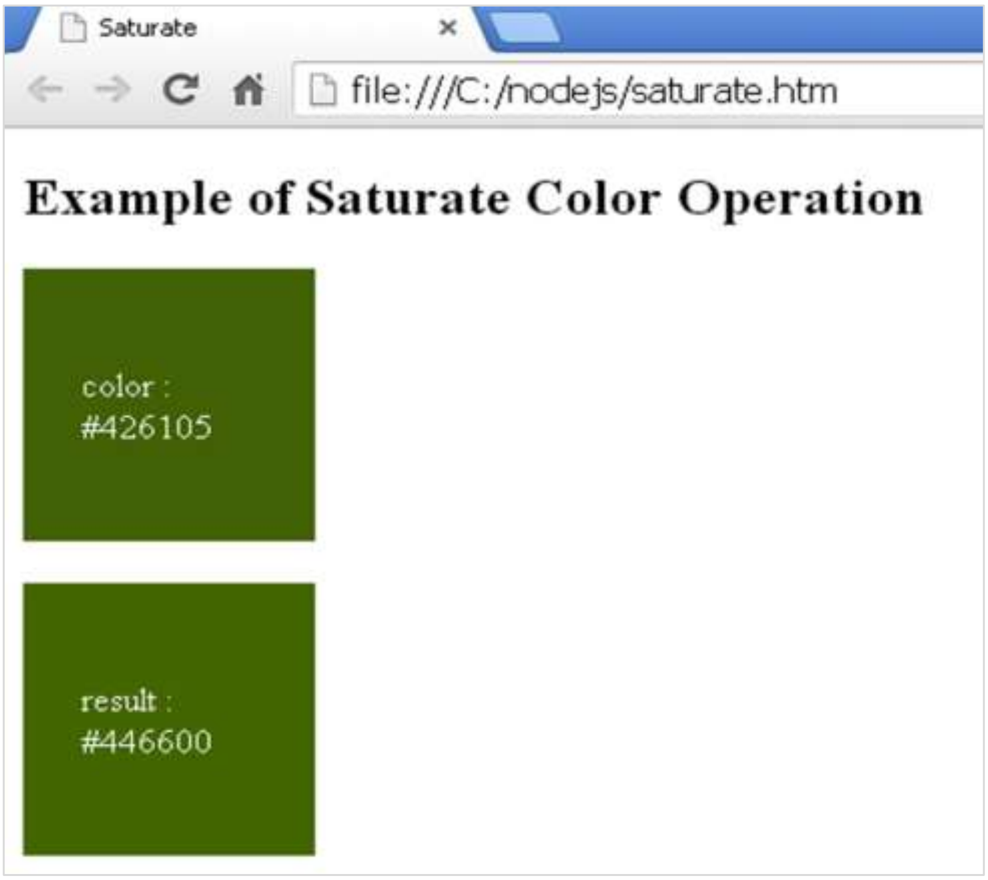
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #426105;
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #446600;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Saturate Color Operation

color : #426105

result : #446600

Less — Desaturate

Description

It decreases the intensity or saturation of a color in the element. It has the following parameters:

- **color**: It represents color object.
- **amount**: It contains percentage between 0 - 100%.
- **method**: It is optional parameter which is used for adjustment to be relative to current value by setting it to *relative*.

Example

The following example demonstrates the use of desaturate color operation in the LESS file:

```
<html>
<head>
   <title>Desaturate</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Desaturate Color Operation</h2>
   <div class="myclass1">
   <p>color :<br> #426105</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br> #415c0a</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsl(80, 90%, 20%);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: desaturate(hsl(80, 90%, 20%), 10%);
   color:white;
}
```

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:

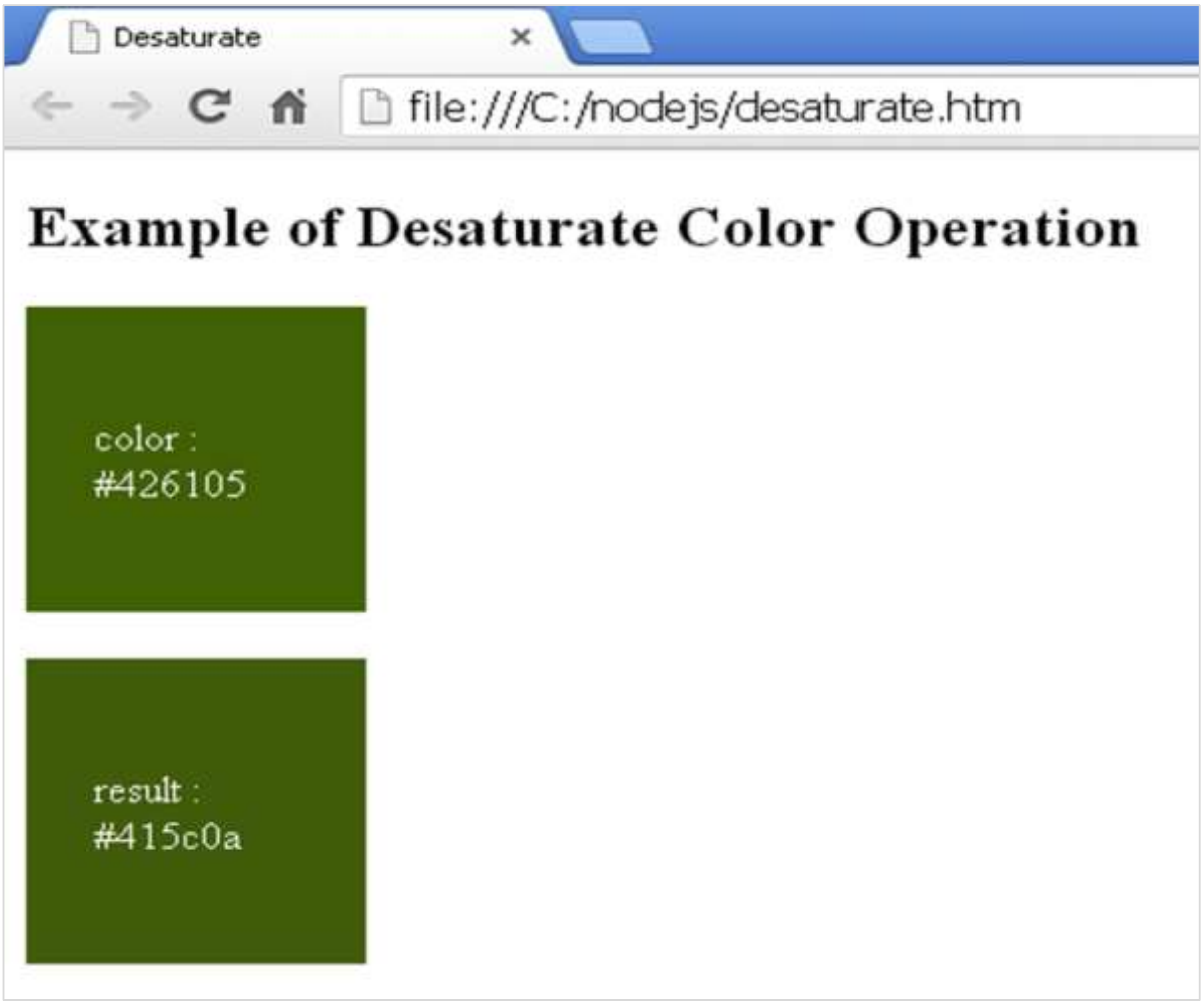
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #426105;
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #415c0a;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Desaturate Color Operation

color : #426105

result : #415c0a

Less — Lighten

Description

It lightens the color in the element. It has the following parameters:

- **color**: It represents the color object.
- **amount**: It contains percentage between 0 - 100%.
- **method**: It is an optional parameter which is used for adjustment to be relative to current value by setting it to *relative*.

Example

The following example demonstrates the use of lighten color operation in the LESS file:

```
<html>
<head>
   <title>Lighten</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Lighten Color Operation</h2>
   <div class="myclass1">
   <p>color :<br> #426105</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br> #639108</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsl(80, 90%, 20%);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: lighten(hsl(80, 90%, 20%), 10%);
   color:white;
}
```

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:

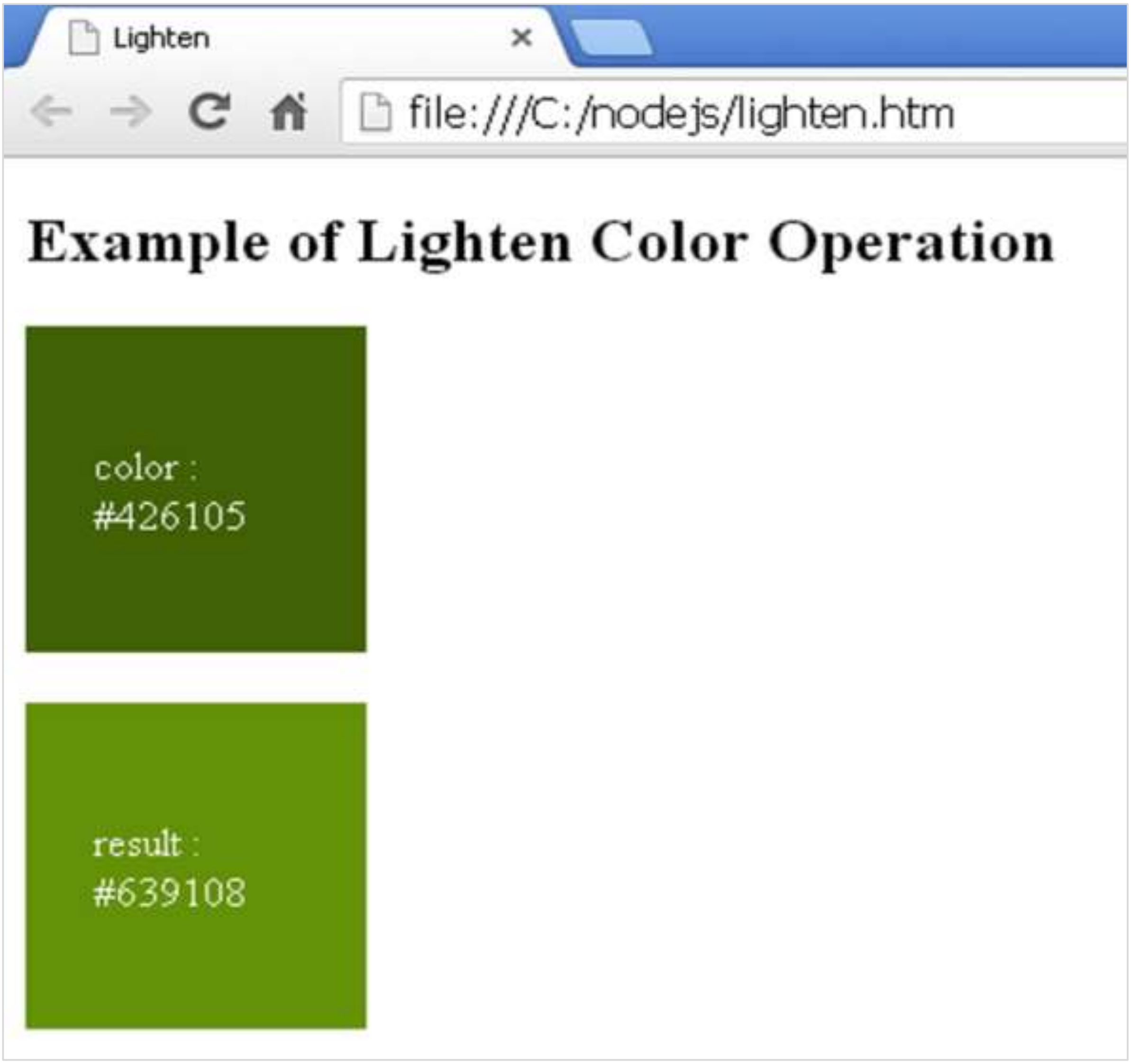
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #426105;
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #639108;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Lighten Color Operation

color : #426105

result : #639108

Less — Darken

Description

It decreases the lightness of a color in the element. It has following parameters:

- **color**: It represents color object.
- **amount**: It contains percentage between 0 - 100%.
- **method**: It is optional parameter which is used for adjustment to be relative to current value by setting it to *relative*.

Example

The following example demonstrates the use of darken color operation in the LESS file:

```
<html>
<head>
   <title>Darken</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Darken Color Operation</h2>
   <div class="myclass1">
   <p>color :<br> #426105</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br> #213003</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsl(80, 90%, 20%);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: darken(hsl(80, 90%, 20%), 10%);
   color:white;
}
```

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:

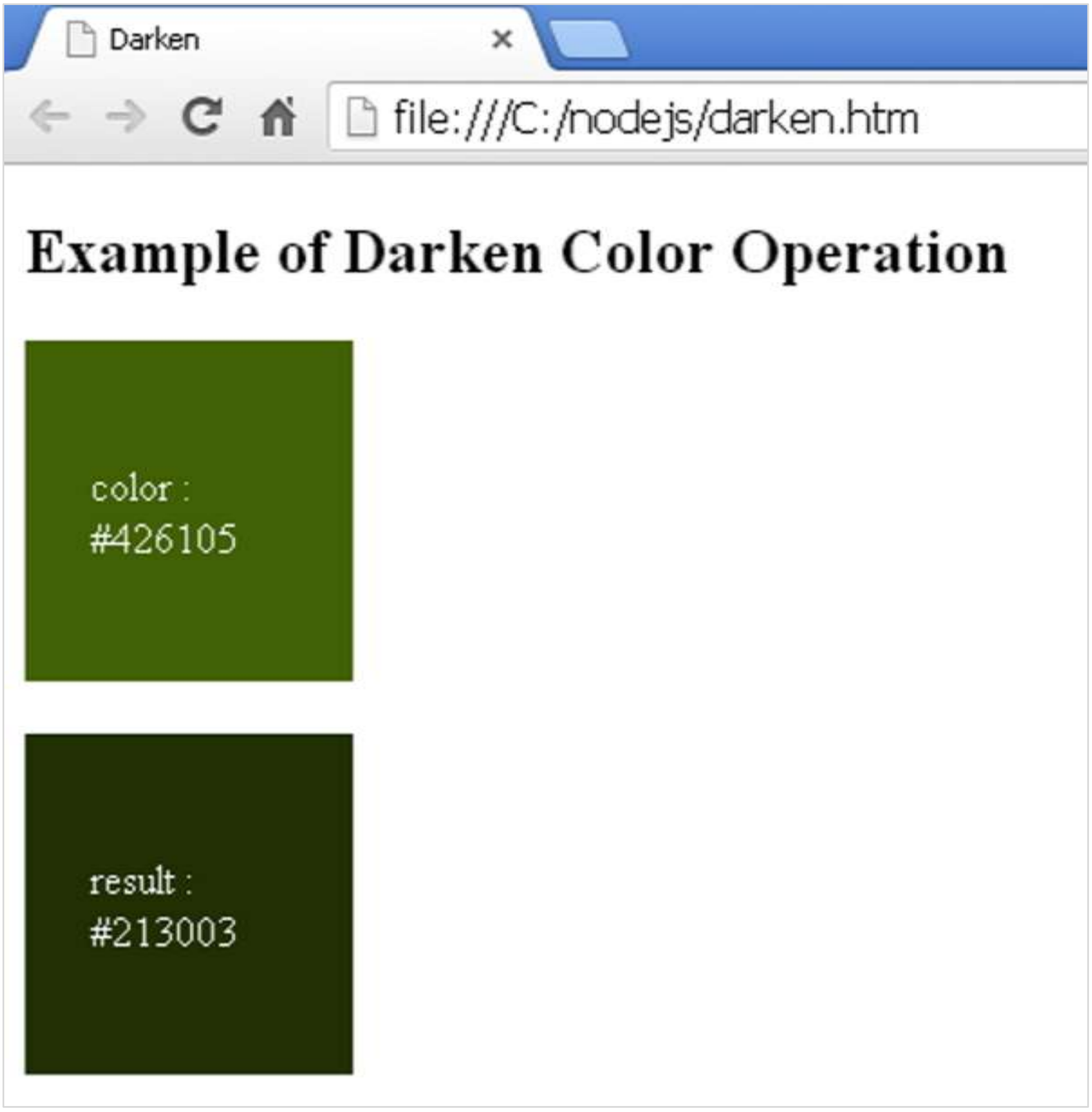
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #426105;
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #213003;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Darken Color Operation

color : #426105

result : #213003

Less — Fadein

Description

It increases the opacity for selected elements. It has following parameters:

- **color**: It represents color object.
- **amount**: It contains percentage between 0 - 100%.
- **method**: It is optional parameter which is used for adjustment to be relative to current value by setting it to *relative*.

Example

The following example demonstrates the use of **fadein** color operation in the LESS file:

```
<html>
<head>
   <title>Fadein</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Fadein Color Operation</h2>
   <div class="myclass1">
   <p>color :<br>rgba(66, 97, 5, 0.5)</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br>rgba(66, 97, 5, 0.6)</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsla(80, 90%, 20%,0.5);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: fadein(hsla(80, 90%, 20%,0.5), 10%);
   color:white;
}
```

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:

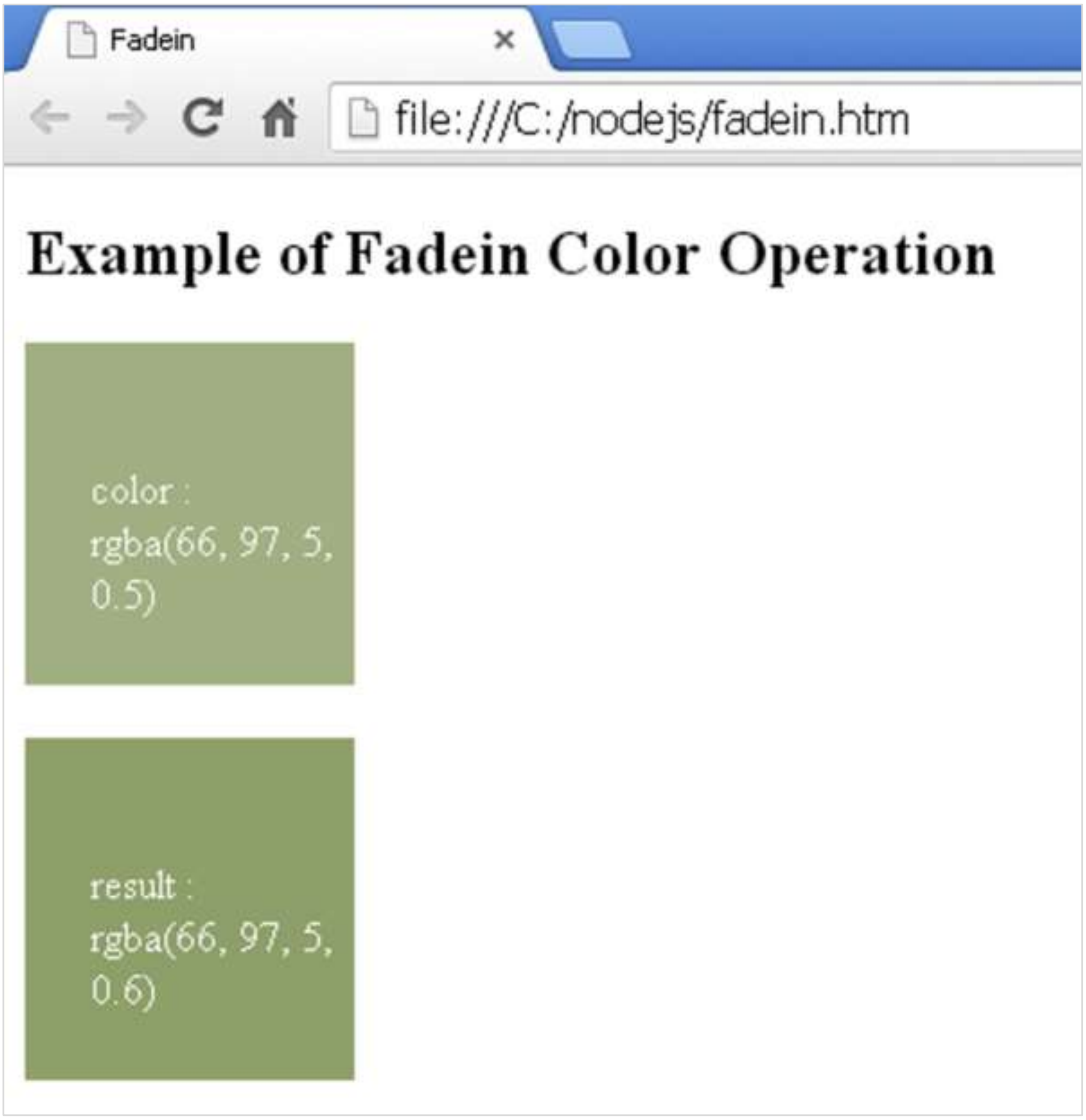
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: rgba(66, 97, 5, 0.5);
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: rgba(66, 97, 5, 0.6);
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Fadein Color Operation

color : rgba(66, 97, 5, 0.5)

result : rgba(66, 97, 5, 0.6)

Less — Fadeout

Description

It decrease the opacity for selected elements. It has following parameters:

- **color**: It represents color object.
- **amount**: It contains percentage between 0 - 100%.
- **method**: It is optional parameter which is used for adjustment to be relative to current value by setting it to *relative*.

Example

The following example demonstrates the use of fadeout color operation in the LESS file:

```
<html>
<head>
   <title>Fadeout</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Fadeout Color Operation</h2>
   <div class="myclass1">
   <p>color :<br>rgba(66, 97, 5, 0.5)</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br>rgba(66, 97, 5, 0.4)</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsla(80, 90%, 20%,0.5);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: fadeout(hsla(80, 90%, 20%,0.5), 10%);
   color:white;
}
```

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:

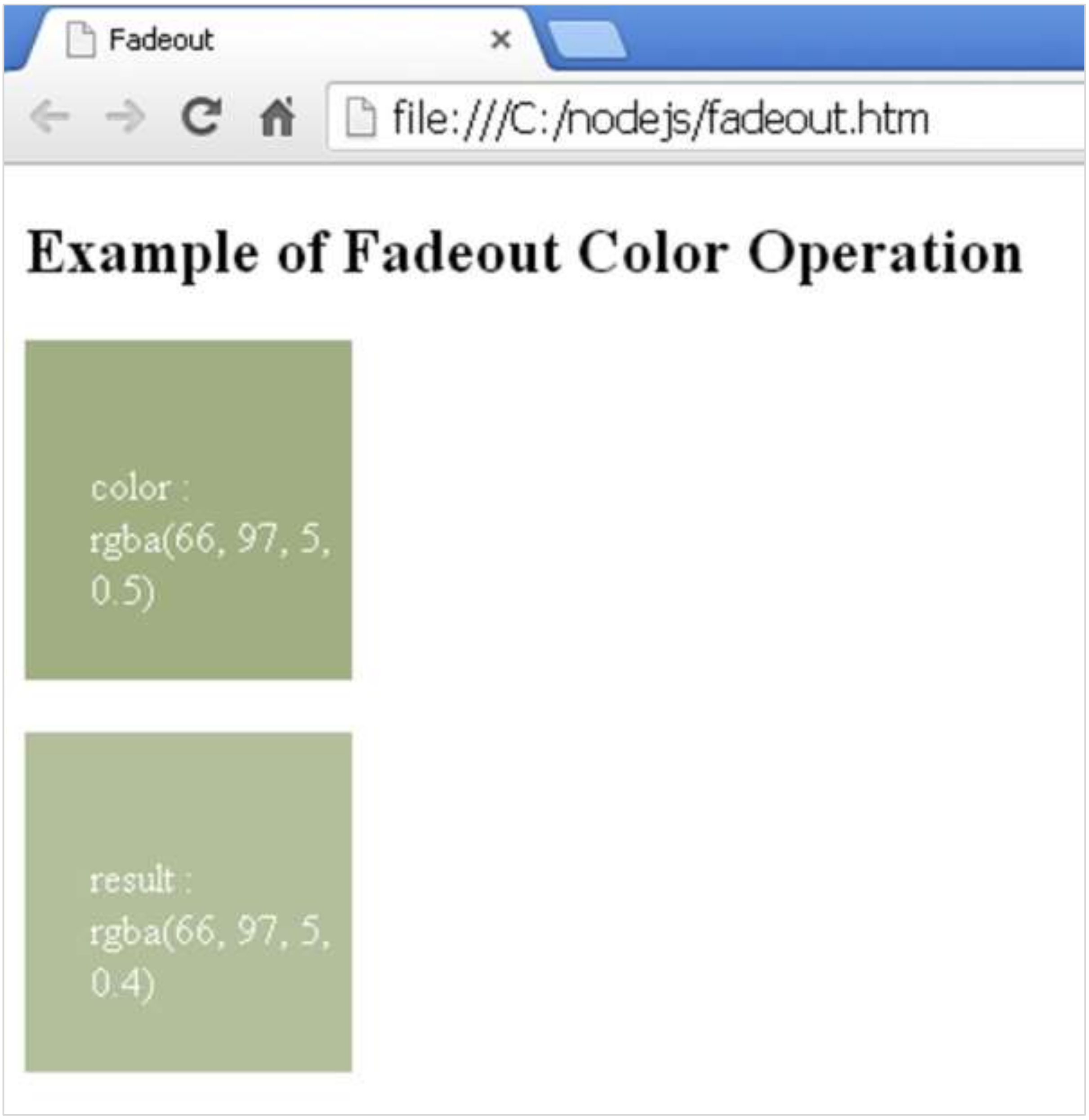
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: rgba(66, 97, 5, 0.5);
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: rgba(66, 97, 5, 0.4);
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Fadeout Color Operation

color : rgba(66, 97, 5, 0.5)

result : rgba(66, 97, 5, 0.4)

Less — Fade

Description

It is used to set the transparency of a color for selected elements. It has following parameters:

- **color**: It represents color object.
- **amount**: It contains percentage between 0 - 100%.

Example

The following example demonstrates the use of fade color operation in the LESS file:

```
<html>
<head>
   <title>Fade</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Fade Color Operation</h2>
   <div class="myclass1">
   <p>color :<br>#426105</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br>rgba(66, 97, 5, 0.1)</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsl(80, 90%, 20%);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: fade(hsl(80, 90%, 20%), 10%);
   color:black;
}
```

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:

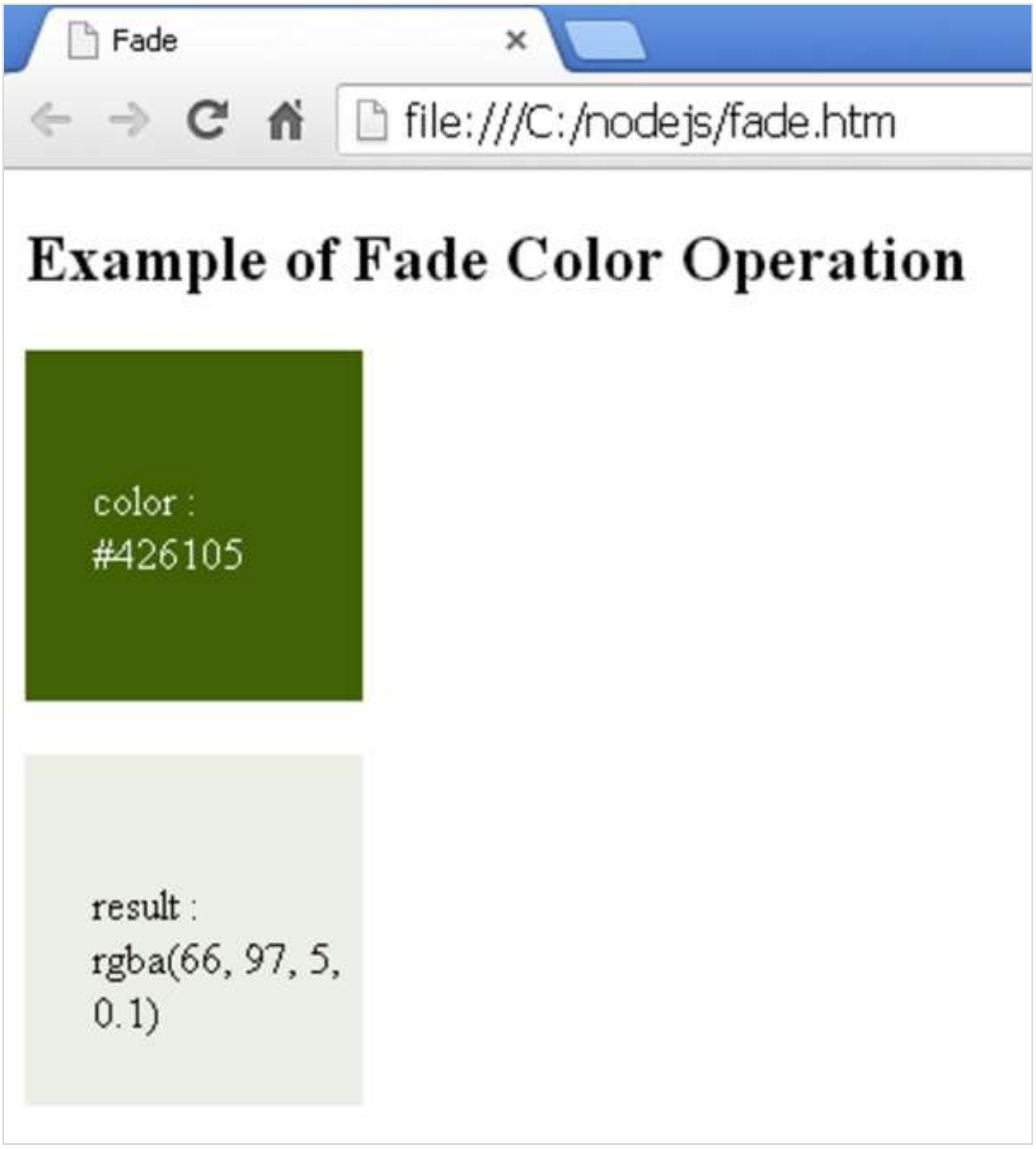
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #426105;
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: rgba(66, 97, 5, 0.1);
  color: black;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Fade Color Operation

```
color :
#426105
```

```
result :
rgba(66, 97, 5,
0.1)
```

Less — Spin

Description

It is used to rotate the angle of a color for selected elements. It has the following parameters:

- **color**: It represents a color object.
- **amount**: It contains percentage between 0 - 100%.

Example

The following example demonstrates the use of spin color operation in the LESS file:

```
<html>
<head>
   <title>Spin</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Spin Color Operation</h2>
   <div class="myclass1">
   <p>color :<br>#426105</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br>#526105</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsl(80, 90%, 20%);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: spin(hsl(80, 90%, 20%), -10);
   color:white;
}
```

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:

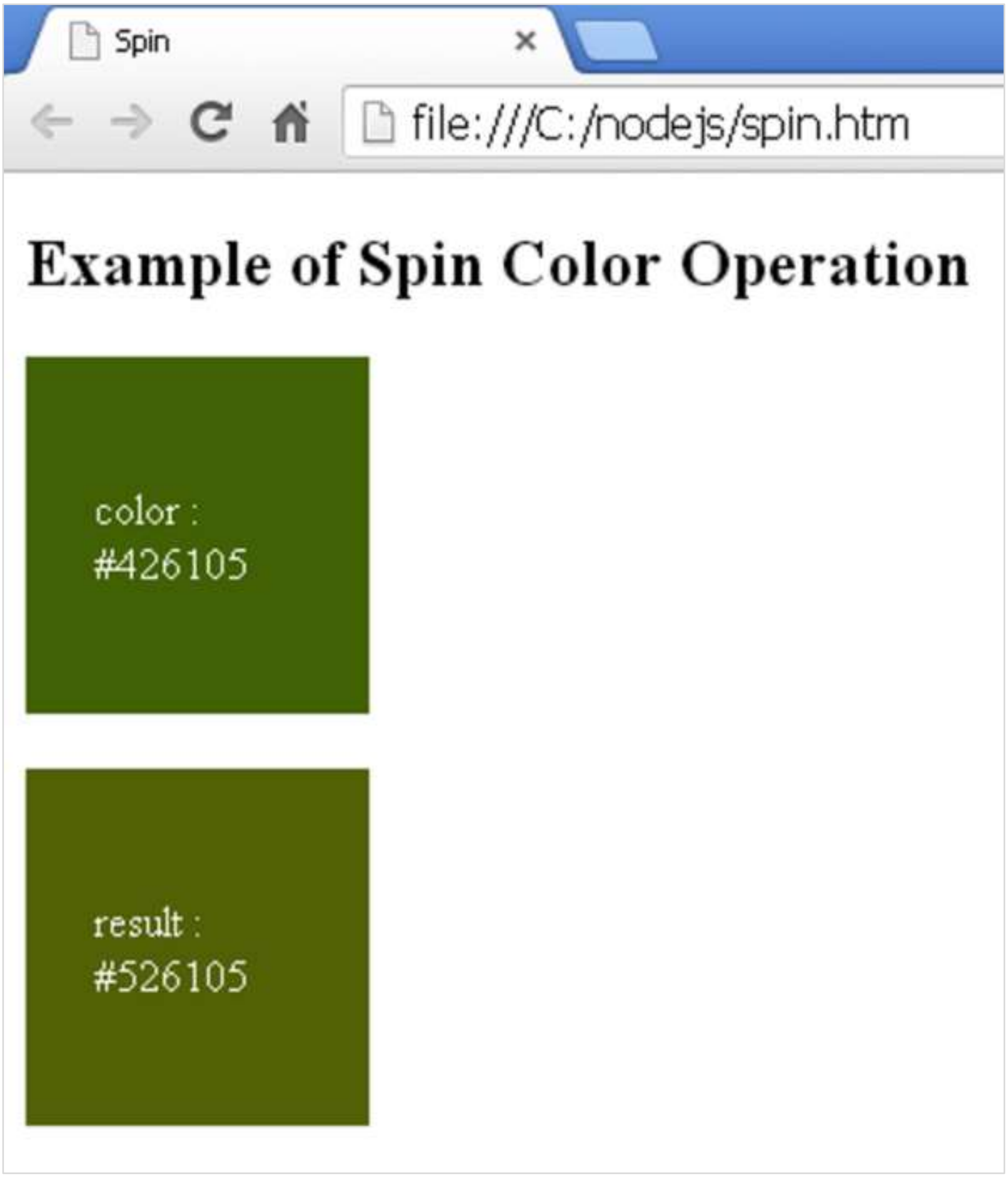
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #426105;
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #526105;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Spin Color Operation

color : #426105

result : #526105

Less — Mix

Description

It is used to mix the two colors along with the opacity. It has the following parameters:

- **color1**: It represents a color object.
- **color2**: It represents a color object.
- **weight**: It is an optional parameter that specifies weight of the element by providing percentage balance point between the two colors.

Example

The following example demonstrates the use of mix color operation in the LESS file:

```
<html>
<head>
   <title>Mix</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Mix Color Operation</h2>
   <div class="myclass">
   <p>Mixed color :<br>#b0897d</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: mix(#b361b1, #acb148, 50%);
   color:white;
}
```

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:

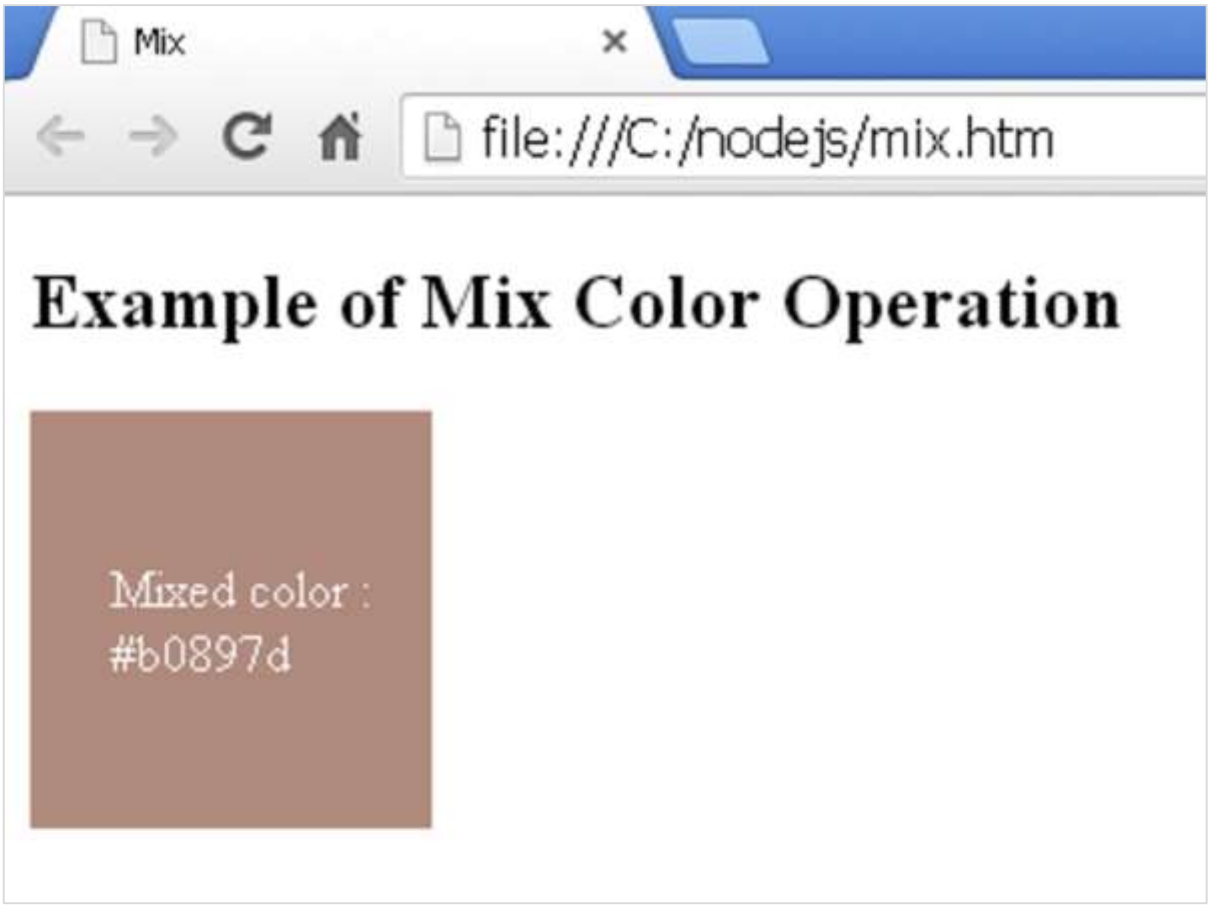
style.css

```
.myclass {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #b0897d;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Mix Color Operation

Mixed color : #b0897d

Less — Tint

Description

It is used to mix the color with white as you decrease the proportion of the color. It has the following parameters:

- **color**: It represents a color object.
- **weight**: It is an optional parameter that specifies the weight of an element by providing percentage balance point between color and white.

Example

The following example demonstrates the use of tint color operation in the LESS file:

```
<html>
<head>
   <title>Tint</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Tint Color Operation</h2>
   <div class="myclass1">
   <p>color :<br>rgba(66, 97, 5, 0.5)</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br>rgba(208, 216, 193, 0.75)</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: rgba(66, 97, 5, 0.5);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: tint(rgba(66, 97, 5, 0.5),50%);
   color:white;
}
```

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:

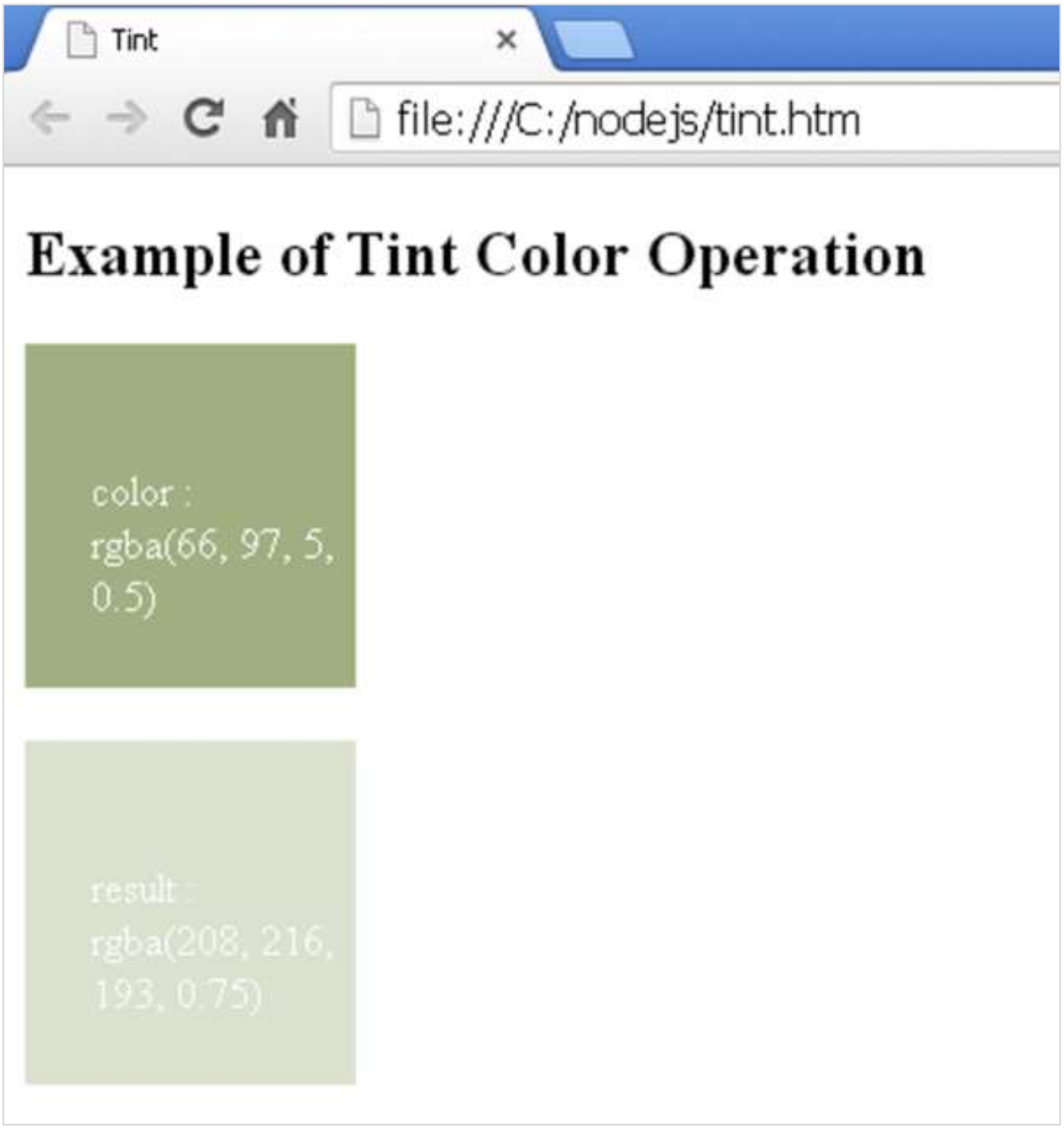
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: rgba(66, 97, 5, 0.5);
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: rgba(208, 216, 193, 0.75);
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Tint Color Operation

color : rgba(66, 97, 5, 0.5)

result : rgba(208, 216, 193, 0.75)

Less — Shade

Description

It is used to mix the color with black as you decrease the proportion of the color. It has the following parameters:

- **color**: It represents a color object.
- **weight**: It is an optional parameter that specifies the weight of the element by providing the percentage balance point between color and black.

Example

The following example demonstrates the use of shade color operation in the LESS file:

```
<html>
<head>
   <title>Shade</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Shade Color Operation</h2>
   <div class="myclass1">
   <p>color :<br>rgba(66, 97, 5, 0.5)</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br>rgba(17, 24, 1, 0.75)</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: rgba(66, 97, 5, 0.5);
   color:white;
}

.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: shade(rgba(66, 97, 5, 0.5),50%);
   color:white;
}
```

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:

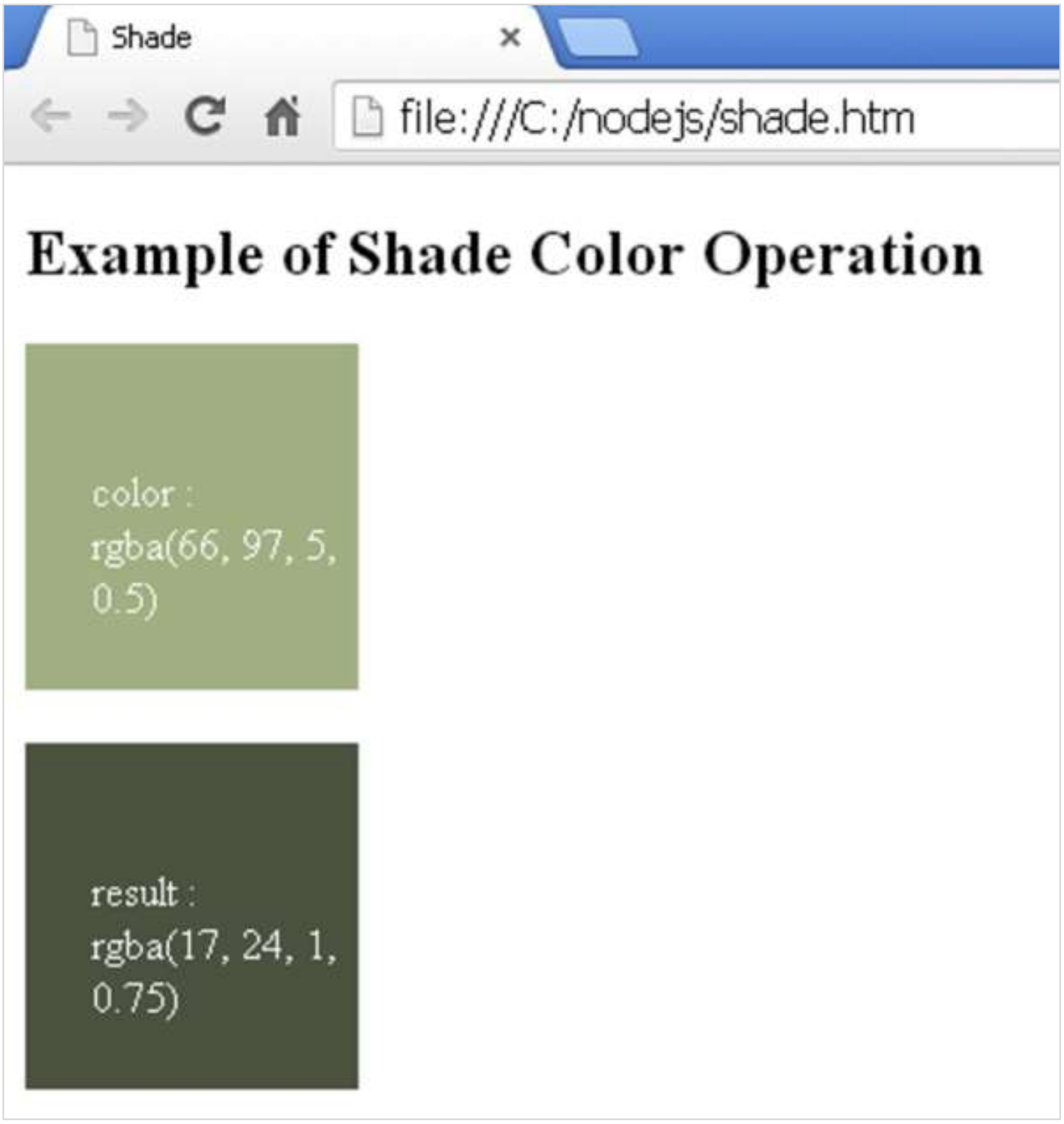
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: rgba(66, 97, 5, 0.5);
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: rgba(17, 24, 1, 0.75);
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Shade Color Operation

color : rgba(66, 97, 5, 0.5)

result : rgba(17, 24, 1, 0.75)

Less — Greyscale

Description

It discards the saturation from a color in the selected elements. It has the following parameters:

- **color**: It represents a color object.

Example

The following example demonstrates the use of greyscale color operation in the LESS file:

```
<html>
<head>
   <title>Greyscale</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Greyscale Color Operation</h2>
   <div class="myclass1">
   <p>color :<br>#426105</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br>#333333</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsl(80, 90%, 20%);
   color:white;
}
```

```
.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: greyscale(hsl(80, 90%, 20%));
   color:white;
}
```

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:

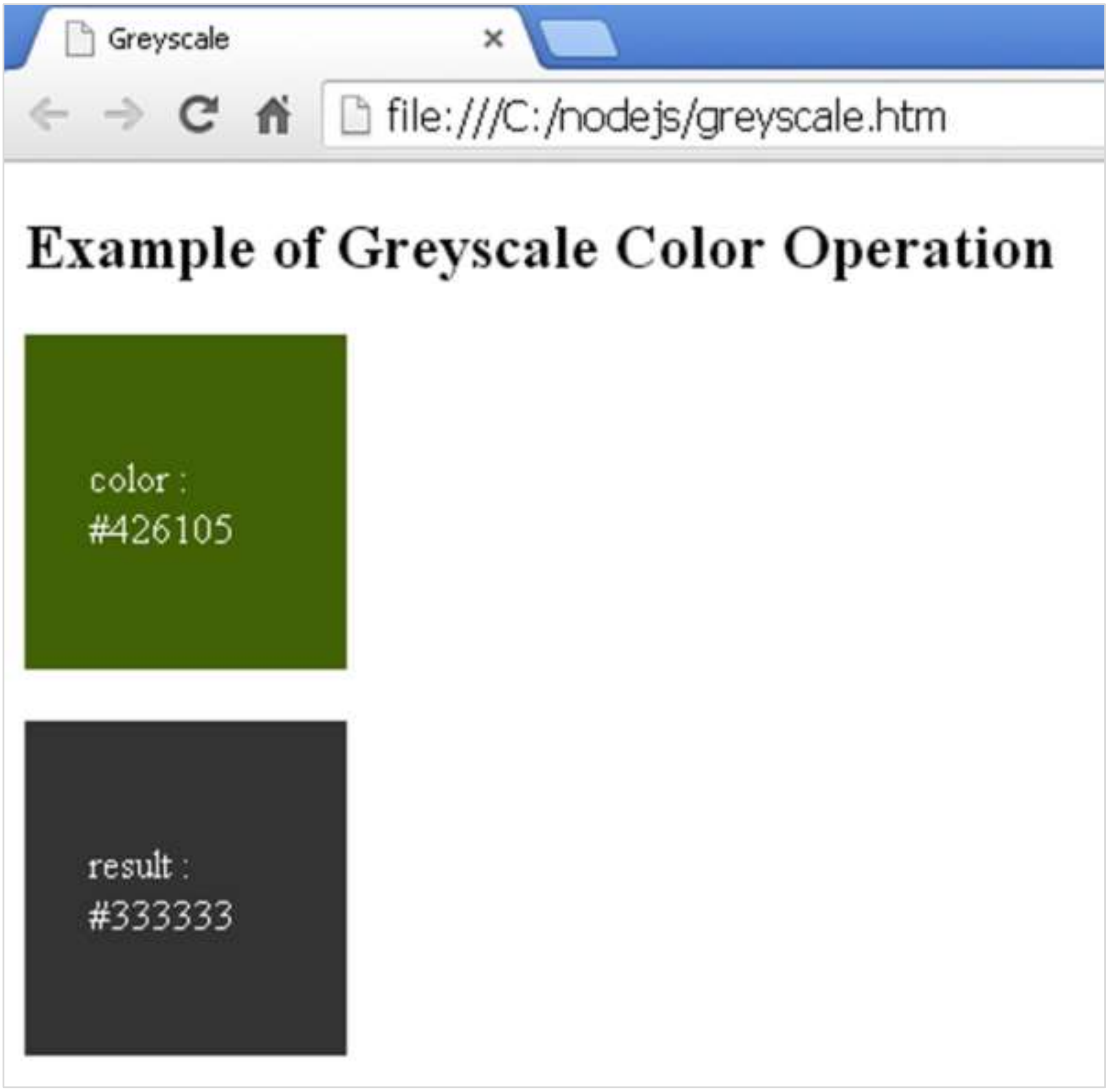
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #426105;
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #333333;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Greyscale Color Operation

color : #426105

result : #333333

Less — Contrast

Description

It sets the contrast for the colors in the element and alters the difference between light and dark values. It has the following parameters:

- **color**: It represents a color object.
- **dark**: It is an optional parameter that sets dark color.
- **light**: It is an optional parameter that sets light color.
- **threshold**: It is an optional parameter that contains percentage between 0 - 100% and specifies transition from dark to light.

Example

The following example demonstrates the use of contrast color operation in the LESS file:

```
<html>
<head>
   <title>Contrast</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <h2>Example of Contrast Color Operation</h2>
   <div class="myclass1">
   <p>color :<br>#426105</p>
   </div><br>

   <div class="myclass2">
      <p>result :<br>#81F79F</p>
   </div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.myclass1{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: hsl(80, 90%, 20%);
   color:white;
}
.myclass2{
   height:100px;
   width:100px;
   padding: 30px 0px 0px 25px;
   background-color: contrast(hsl(80, 90%, 20%), #81F79F, #01DFA5, 30%);
   color:white;
}
```

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:

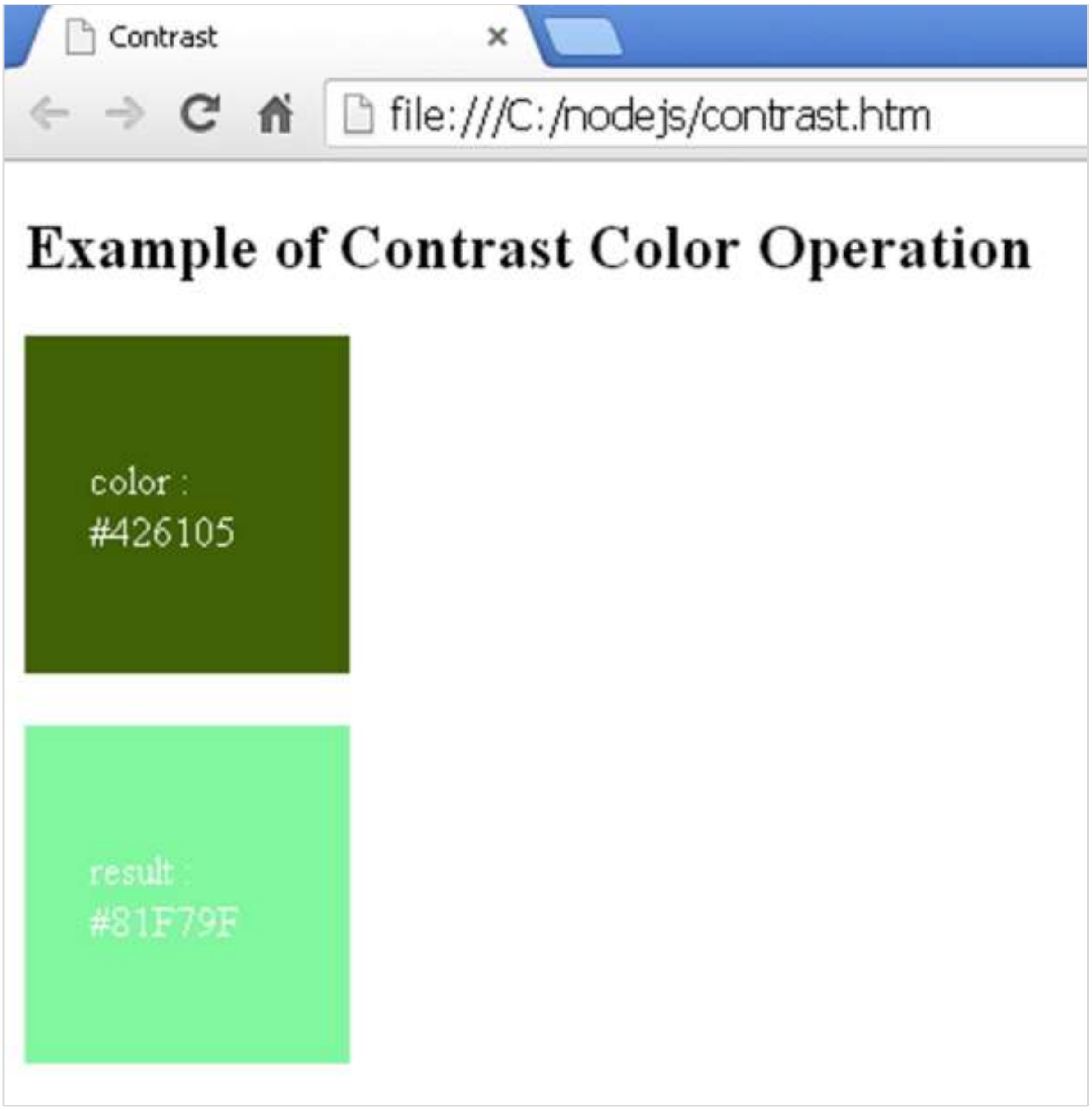
style.css

```
.myclass1 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #426105;
  color: white;
}
.myclass2 {
  height: 100px;
  width: 100px;
  padding: 30px 0px 0px 25px;
  background-color: #81F79F;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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



Example of Contrast Color Operation

color :
#426105

result :
#81F79F

33. Less – Color Blending Functions

In this chapter, we will understand the **Color Blending Functions** in LESS. These are similar operations used in image editors like Photoshop, Fireworks or GIMP, which matches your CSS colors to your images.

The following table shows the color blending functions used in LESS.

S.NO.	Functions & Description
1	**multiply** It multiplies two colors and returns a resultant color.
2	**screen** It takes two colors and returns a brighter color. It works opposite of *multiply* function.
3	**overlay** It generates result by combining the effect of *multiply* and *screen*.
4	**softlight** It works similar to *overlay* but it uses only a part of the color, which soft-highlights the other color.
5	**hardlight** It works similar to *overlay* but the role of the colors reversed.
6	**difference** It subtracts the second input color from the first input color.
7	**exclusion** It works similar to *difference* function but with lower contrast.
8	**average** It computes the average of two input colors on a per-channel (RGB) basis.
9	**negation** It works opposite to *difference* function, which subtracts first color from second color.

Less — Color Blending Multiply Function

Description

The *multiply* function multiplies two colors. The two colors corresponding RGB channels are multiplied together then divided by 255 to get a darker col.

Parameters:

- **color1:** A color object.
- **color2:** A color object.

Returns: color.

Example

The following example demonstrates the use of the *multiply* function in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Multiply Function</h2>
<div class="color1">
  <p>(color1) <br> #ff6600</p>
</div><br>
<div class="color2">
  <p>(color2) <br> #0000ff</p>
</div><br>
<div class="res">
  <p>(result) <br> #000000</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.color1 {
    width: 100px;
    height: 100px;
    background-color: #ff6600;
}
.color2 {
    width: 100px;
    height: 100px;
    background-color: #0000ff;
}
.res {
    width: 100px;
    height: 100px;
    background-color: multiply(#ff6600, #0000ff);
}
p{
 padding: 30px 0px 0px 25px;
 color: white;
}
```

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:

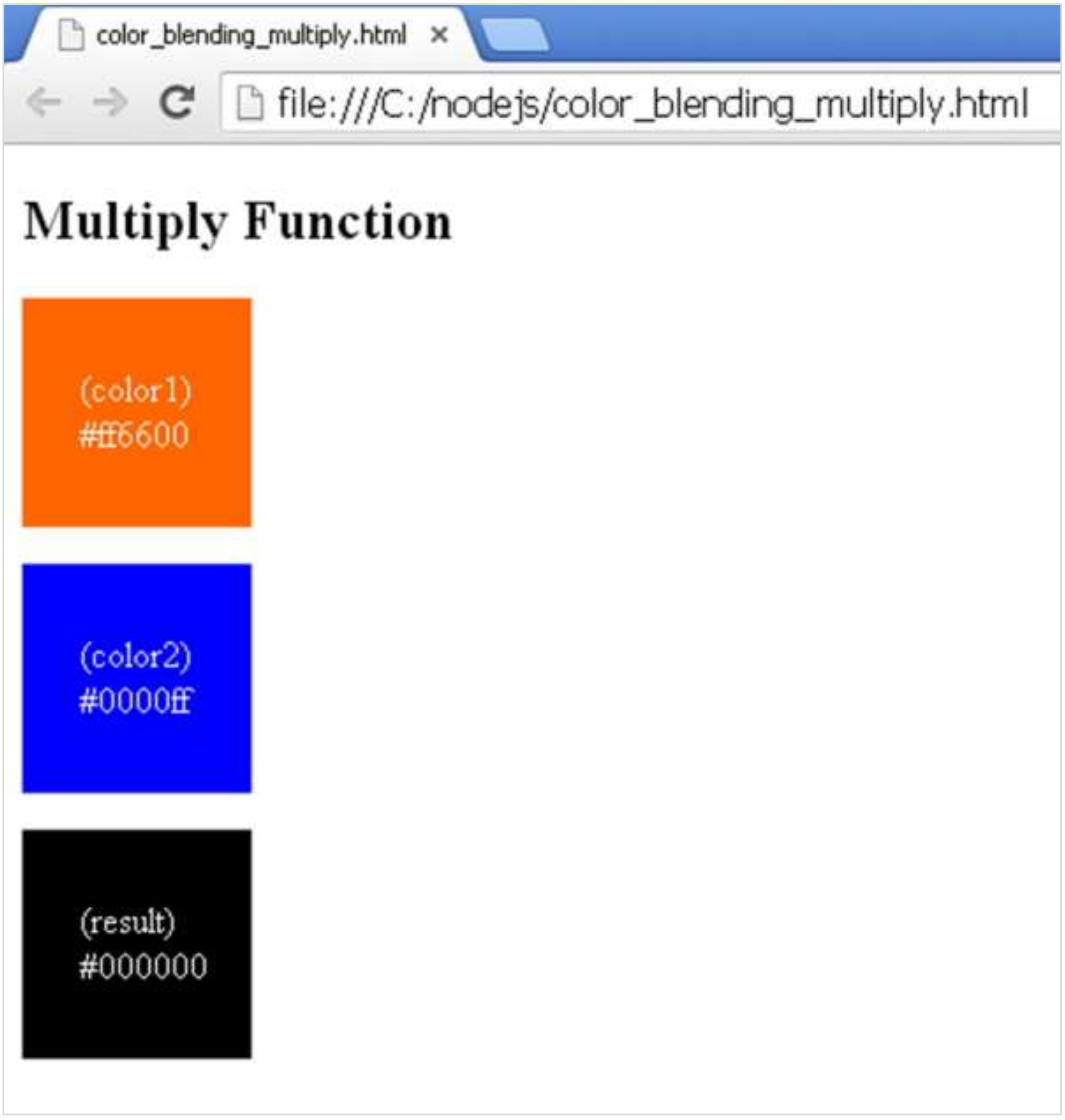
style.css

```
.color1 {
  width: 100px;
  height: 100px;
  background-color: #ff6600;
}
.color2 {
  width: 100px;
  height: 100px;
  background-color: #0000ff;
}
.result {
  width: 100px;
  height: 100px;
  background-color: #000000;
}
p {
  padding: 30px 0px 0px 25px;
  color: white;
}
```

Output

Follow these steps to see how the above code works:

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


color_blending_multiply.html
file:///C:/nodejs/color_blending_multiply.html

Multiply Function

(color1)
#ff6600

(color2)
#0000ff

(result)
#000000

Less — Color Blending Screen Function

Description

The *screen* function takes two colors as parameter and generates a brighter color as output.

Parameters

- **color1:** A color object.
- **color2:** A color object.

Returns: color.

Example

The following example demonstrates the use of *screen* function in a LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Screen Function</h2>
<div class="color1">
  <p>(color1) <br> #ff6600</p>
</div><br>
<div class="color2">
  <p>(color2) <br> #0000ff</p>
</div><br>
<div class="res">
  <p>(result) <br> #ff66ff</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.color1 {
    width: 100px;
    height: 100px;
    background-color: #ff6600;
}
.color2 {
    width: 100px;
    height: 100px;
    background-color: #0000ff;
}
.res {
    width: 100px;
    height: 100px;
    background-color: screen(#ff6600, #0000ff);
}
p{
 padding: 30px 0px 0px 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:

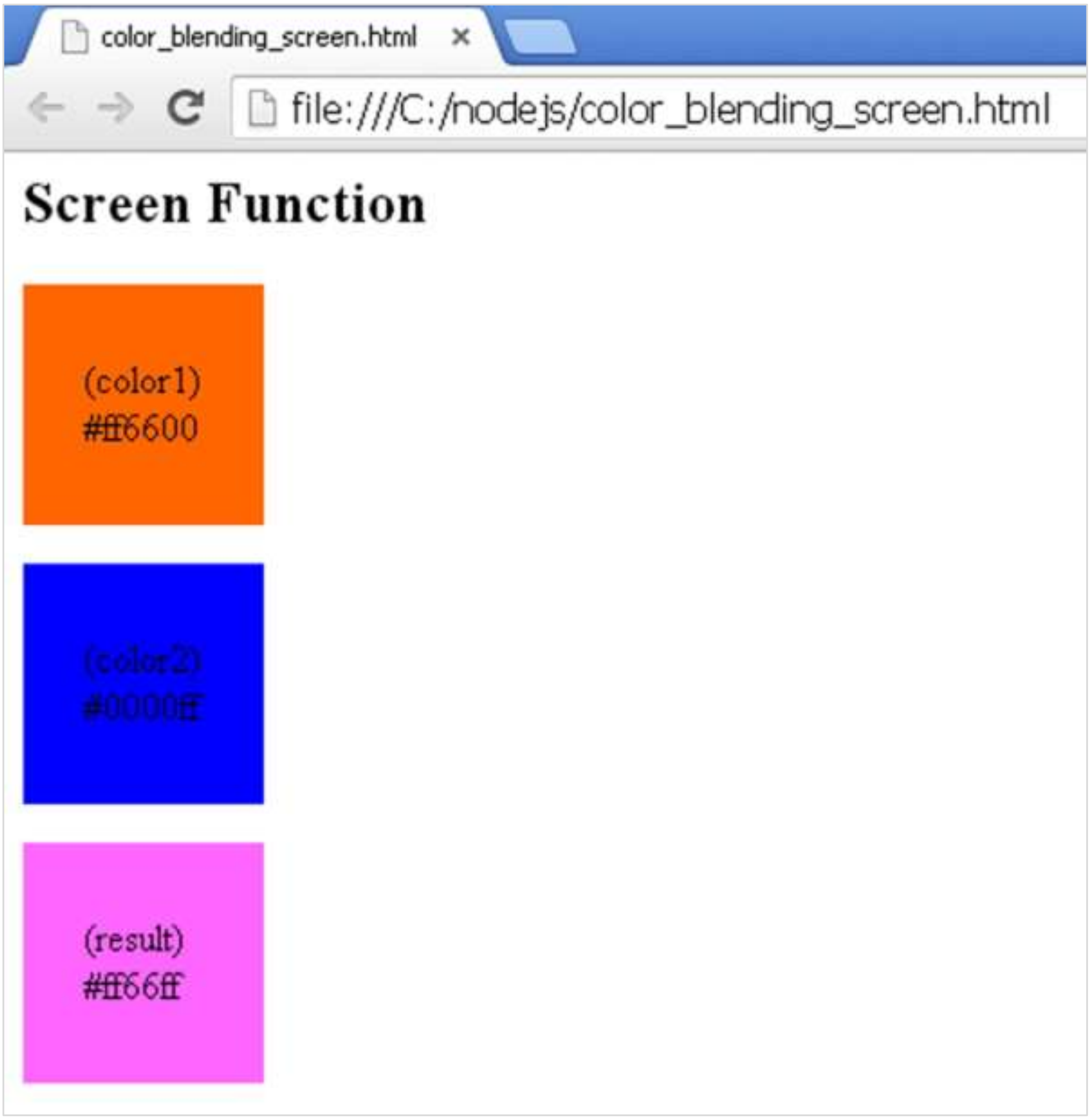
style.css

```
.color1 {
  width: 100px;
  height: 100px;
  background-color: #ff6600;
}
.color2 {
  width: 100px;
  height: 100px;
  background-color: #0000ff;
}
.result {
  width: 100px;
  height: 100px;
  background-color: #ff66ff;
}
p {
  padding: 30px 0px 0px 25px;
}
```

Output

Follow these steps to see how the above code works:

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


color_blending_screen.html
file:///C:/nodejs/color_blending_screen.html

Screen Function

(color1)
#ff6600

(color2)
#0000ff

(result)
#ff66ff

Less — Color Blending Overlay Function

Description

The *overlay* generates result by combining the effect of *multiply* and *screen*. It makes light channels lighter and dark channels darker. The result is based on the first color parameter.

Parameters

- **color1:** A base color object which is the determinant color making the resultant color lighter or darker.
- **color2:** A color object to overlay.

Returns: color.

Example

The following example demonstrates the use of *overlay* function in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Overlay Function</h2>
<div class="color1">
  <p>(color1) <br> #ff6600</p>
</div><br>
<div class="color2">
  <p>(color2) <br> #0000ff</p>
</div><br>
<div class="res">
  <p>(result) <br> #ff0000</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.color1 {
    width: 100px;
    height: 100px;
    background-color: #ff6600;
}
.color2 {
    width: 100px;
    height: 100px;
    background-color: #0000ff;
}
.res {
    width: 100px;
    height: 100px;
    background-color: overlay(#ff6600, #0000ff);
}
p{
 padding: 30px 0px 0px 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:

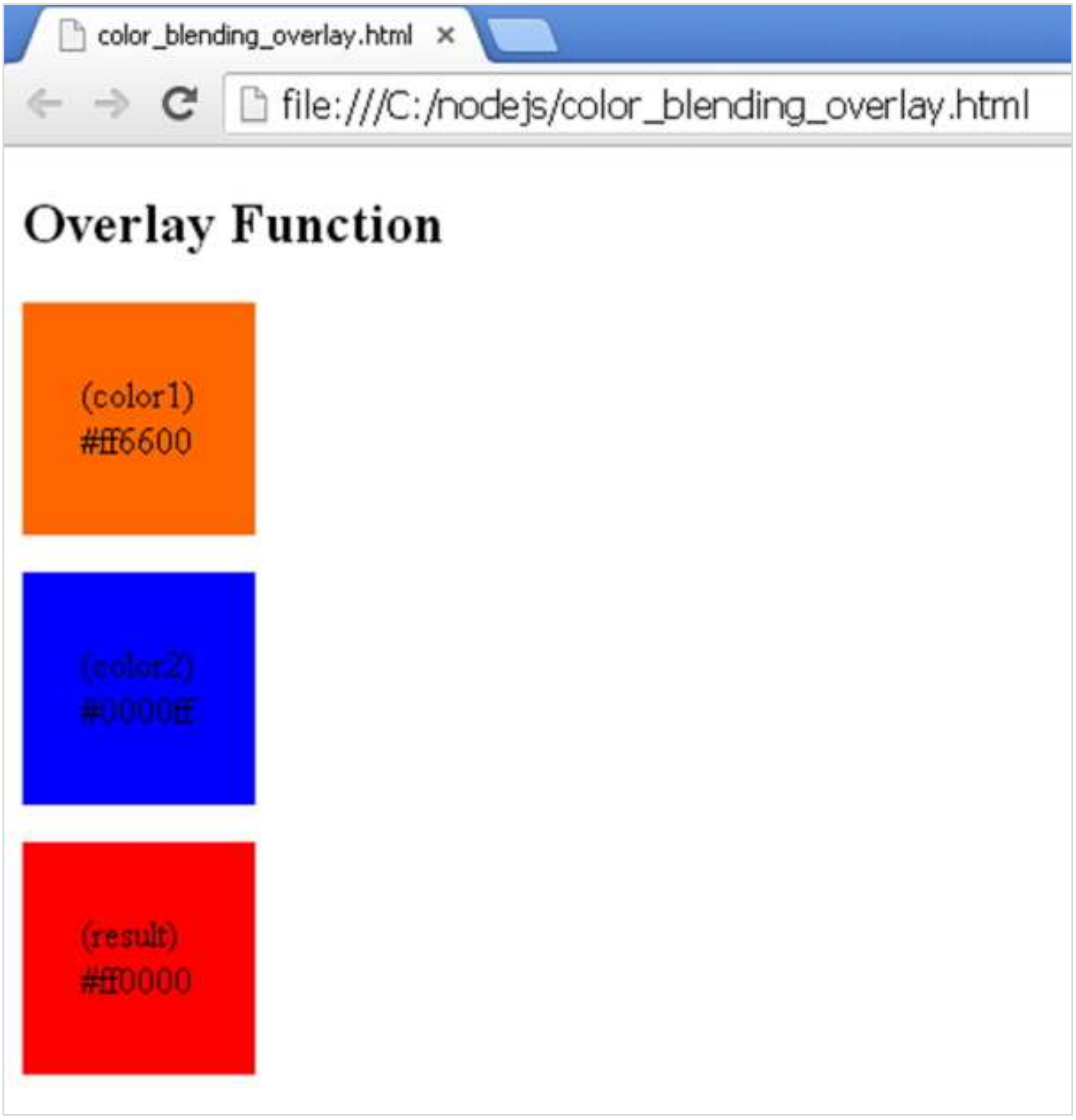
style.css

```
.color1 {
  width: 100px;
  height: 100px;
  background-color: #ff6600;
}
.color2 {
  width: 100px;
  height: 100px;
  background-color: #0000ff;
}
.result {
  width: 100px;
  height: 100px;
  background-color: #ff0000;
}
p {
  padding: 30px 0px 0px 25px;
}
```

Output

Follow these steps to see how the above code works:

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


color_blending_overlay.html
file:///C:/nodejs/color_blending_overlay.html

Overlay Function

(color1)
#ff6600

(color2)
#0000ff

(result)
#ff0000

Less – Color Blending Softlight Function

Description

The *softlight* works similar to *overlay* but it uses only a part of the color, which soft-highlights the other color.

Parameters

- **color1:** A color object to *soft light* another.
- **color2:** A color object that has to be *soft lighten*.

Returns: color.

Example

The following example demonstrates the use of *softlight* function in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Softlight Function</h2>
<div class="color1">
  <p>(color1) <br> #ff6600</p>
</div><br>
<div class="color2">
  <p>(color2) <br> #0000ff</p>
</div><br>
<div class="res">
  <p>(result) <br> #ff2900</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.color1 {
    width: 100px;
    height: 100px;
    background-color: #ff6600;
}
.color2 {
    width: 100px;
    height: 100px;
    background-color: #0000ff;
}
.res {
    width: 100px;
```

```
     height: 100px;
     background-color: softlight(#ff6600, #0000ff);
}
p{
 padding: 30px 0px 0px 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:

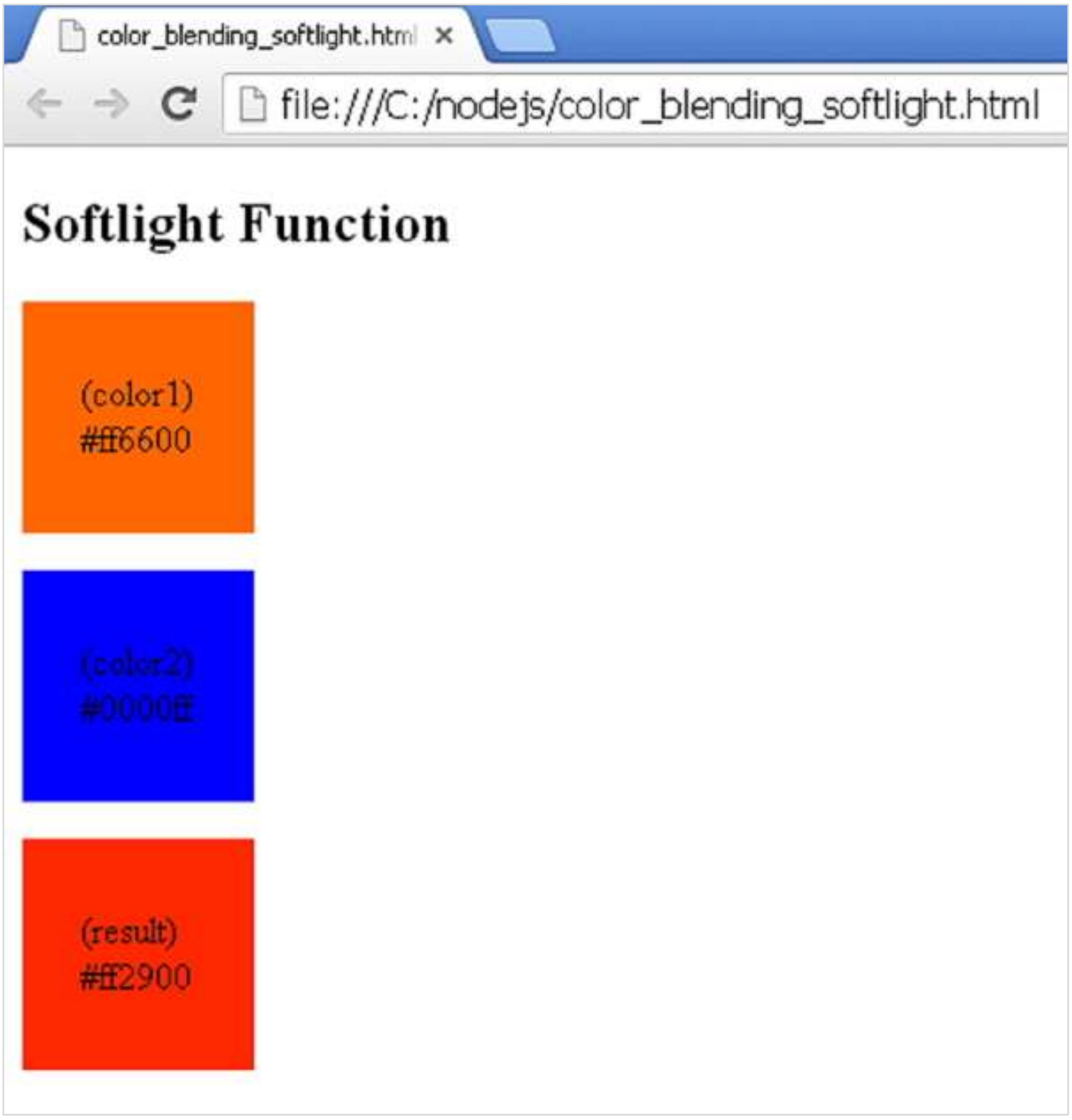
style.css

```
.color1 {
  width: 100px;
  height: 100px;
  background-color: #ff6600;
}
.color2 {
  width: 100px;
  height: 100px;
  background-color: #0000ff;
}
.result {
  width: 100px;
  height: 100px;
  background-color: #ff2900;
}
p {
  padding: 30px 0px 0px 25px;
}
```

Output

Follow these steps to see how the above code works:

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


color_blending_softlight.html
file:///C:/nodejs/color_blending_softlight.html

Softlight Function

(color1)
#ff6600

(color2)
#0000ff

(result)
#ff2900

Less — Color Blending Hardlight Function

Description

The *hardlight* function works similar to *overlay* but the role of the colors reversed. It performs an *overlay()* function with the second parameter to determine whether a multiply or screen operation should be done.

Parameters

- **color1:** A color object to *overlay*.
- **color2:** A base color object which is the determinant color making the resultant color lighter or darker.

Returns: color.

Example

The following example demonstrates the use of *hardlight* function in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Hardlight Function</h2>
<div class="color1">
  <p>(color1) <br> #ff6600</p>
</div><br>
<div class="color2">
  <p>(color2) <br> #0000ff</p>
</div><br>
<div class="res">
  <p>(result) <br> #0000ff</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.color1 {
    width: 100px;
    height: 100px;
    background-color: #ff6600;
}
.color2 {
    width: 100px;
    height: 100px;
    background-color: #0000ff;
}
.res {
    width: 100px;
    height: 100px;
    background-color: hardlight(#ff6600, #0000ff);
}
p{
 padding: 30px 0px 0px 25px;
 color: white;
}
```

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:

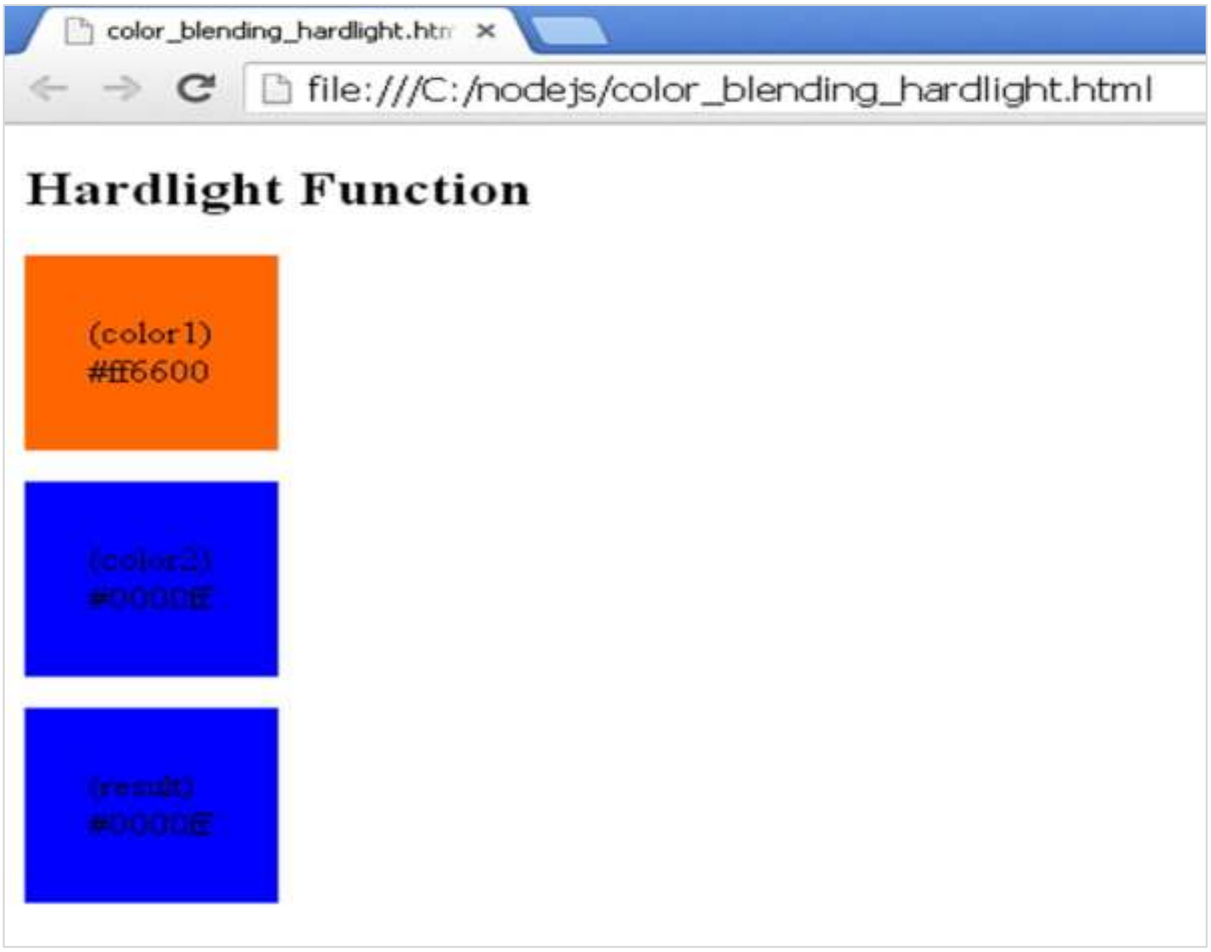
style.css

```
.color1 {
  width: 100px;
  height: 100px;
  background-color: #ff6600;
}
.color2 {
  width: 100px;
  height: 100px;
  background-color: #0000ff;
}
.result {
  width: 100px;
  height: 100px;
  background-color: #0000ff;
}
p {
  padding: 30px 0px 0px 25px;
}
```

Output

Follow these steps to see how the above code works:

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


color_blending_hardlight.htm
file:///C:/nodejs/color_blending_hardlight.html

Hardlight Function

(color1)
#ff6600

(color2)
#0000ff

(result)
#0000ff

Less — Color Blending Difference Function

Description

The *difference* function subtracts the second input color from the first input color on a channel-by-channel basis (note that the negative values are inverted). Subtracting *black* color will result in no change; when *white* color is subtracted, it results in color inversion.

Parameters

- **color1:** A color object which act as *minuend.*
- **color2:** A color object which act as *subtrahend*.

Returns: color.

Example

The following example demonstrates the use of *difference* function in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Difference Function</h2>
<div class="color1">
  <p>(color1) <br> #ff6600</p>
</div><br>
<div class="color2">
  <p>(color2) <br> #333333</p>
</div><br>
<div class="res">
  <p>(result) <br> #cc3333</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.color1 {
    width: 100px;
    height: 100px;
    background-color: #ff6600;
}
.color2 {
    width: 100px;
    height: 100px;
    background-color: #333333;
}
.res {
    width: 100px;
    height: 100px;
    background-color: difference(#ff6600, #333333);
}
p{
 padding: 30px 0px 0px 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:

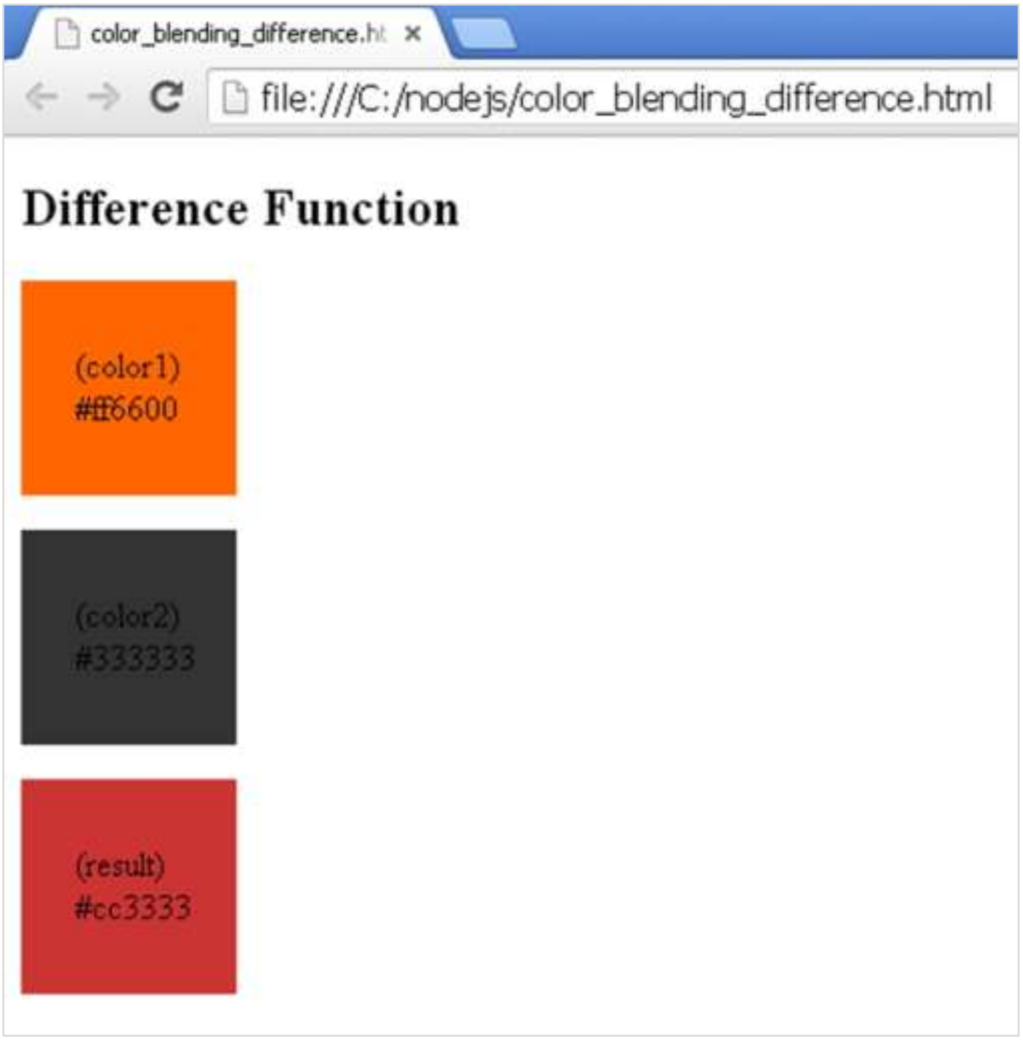
style.css

```
.color1 {
  width: 100px;
  height: 100px;
  background-color: #ff6600;
}
.color2 {
  width: 100px;
  height: 100px;
  background-color: #333333;
}
.result {
  width: 100px;
  height: 100px;
  background-color: #cc3333;
}
p {
  padding: 30px 0px 0px 25px;
}
```

Output

Follow these steps to see how the above code works:

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


color_blending_difference.ht
file:///C:/nodejs/color_blending_difference.html

Difference Function

(color1)
#ff6600

(color2)
#333333

(result)
#cc3333

Less — Color Blending Exclusion Function

Description

The *exclusion* function works similar to *difference* function but with lower contrast.

Parameters

- **color1:** A color object which act as *minuend.*
- **color2:** A color object which act as *subtrahend*.

Returns: color.

Example

The following example demonstrates the use of *exclusion* function in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Exclusion Function</h2>
<div class="color1">
  <p>(color1) <br> #ff6600</p>
</div><br>
<div class="color2">
  <p>(color2) <br> #333333</p>
</div><br>
<div class="res">
  <p>(result) <br> #cc7033</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.color1 {
    width: 100px;
    height: 100px;
    background-color: #ff6600;
}
.color2 {
    width: 100px;
    height: 100px;
    background-color: #333333;
}
.res {
    width: 100px;
    height: 100px;
    background-color: exclusion(#ff6600, #333333);
}
p{
 padding: 30px 0px 0px 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:

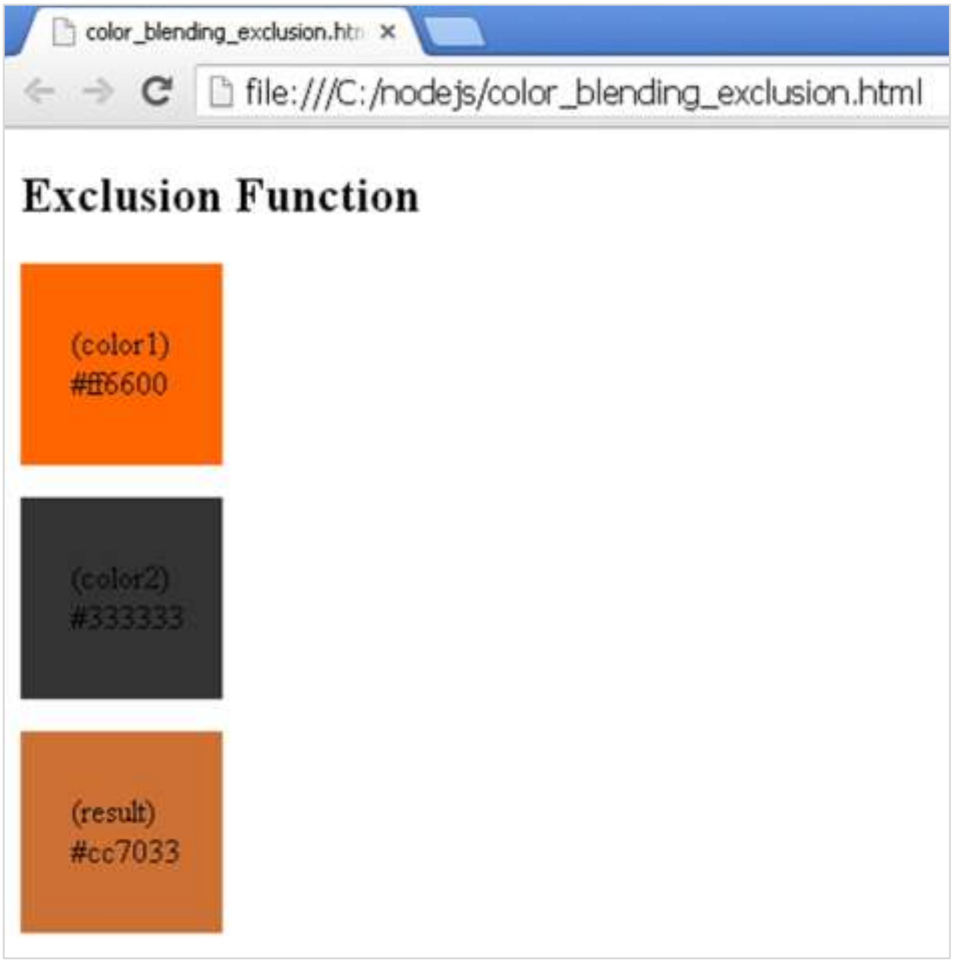
style.css

```
.color1 {
  width: 100px;
  height: 100px;
  background-color: #ff6600;
}
.color2 {
  width: 100px;
  height: 100px;
  background-color: #333333;
}
.result {
  width: 100px;
  height: 100px;
  background-color: #cc7033;
}
p {
  padding: 30px 0px 0px 25px;
}
```

Output

Follow these steps to see how the above code works:

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



color_blending_exclusion.htm
file:///C:/nodejs/color_blending_exclusion.html

Exclusion Function

(color1)
#ff6600

(color2)
#333333

(result)
#cc7033

Less — Color Blending Average Function

Description

The *average* function computes the average of two input colors on a per-channel (RGB) basis.

Parameters

- **color1:** A color object.
- **color2:** A color object.

Returns: color.

Example

The following example demonstrates the use of *average* function in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Average Function</h2>
<div class="color1">
  <p>(color1) <br> #ff6600</p>
</div><br>
<div class="color2">
  <p>(color2) <br> #0000ff</p>
</div><br>
<div class="res">
  <p>(result) <br> #803380</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.color1 {
    width: 100px;
    height: 100px;
    background-color: #ff6600;
}
.color2 {
    width: 100px;
    height: 100px;
    background-color: #0000ff;
}
.res {
    width: 100px;
    height: 100px;
    background-color: average(#ff6600, #0000ff);
}
p{
 padding: 30px 0px 0px 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:

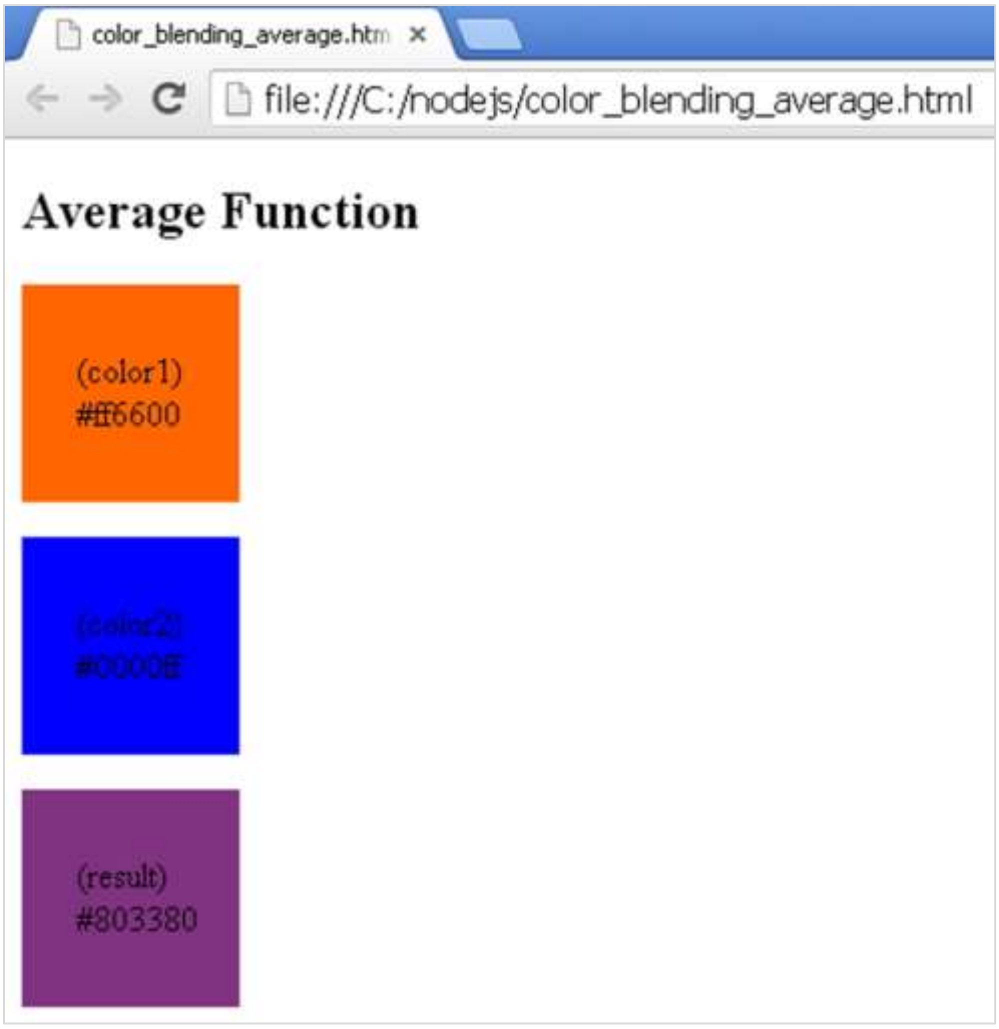
style.css

```
.color1 {
  width: 100px;
  height: 100px;
  background-color: #ff6600;
}
.color2 {
  width: 100px;
  height: 100px;
  background-color: #0000ff;
}
.result {
  width: 100px;
  height: 100px;
  background-color: #803380;
}
p {
  padding: 30px 0px 0px 25px;
}
```

Output

Follow these steps to see how the above code works:

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


color_blending_average.htm
file:///C:/nodejs/color_blending_average.html
Average Function
(color1)
#ff6600
(color2)
#0000ff
(result)
#803380

Less — Color Blending Negation Function

Description

The *negation* function has opposite effect to *difference* function, which subtracts the first input color from the second input color. It results in brighter color. The negative values are inverted.

Parameters

- **color1:** A color object which act as *minuend.*
- **color2:** A color object which act as *subtrahend*.

Returns: color.

Example

The below example demonstrates the use of *negation* function in the LESS file:

```
<html>
<head>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<h2>Negation Function</h2>
<div class="color1">
  <p>(color1) <br> #ff6600</p>
</div><br>
<div class="color2">
  <p>(color2) <br> #333333</p>
</div><br>
<div class="res">
  <p>(result) <br> #cc9933</p>
</div>
</body>
</html>
```

Next, create the *style.less* file.

style.less

```
.color1 {
    width: 100px;
    height: 100px;
    background-color: #ff6600;
}
.color2 {
    width: 100px;
    height: 100px;
    background-color: #333333;
}
.res {
    width: 100px;
    height: 100px;
    background-color: negation(#ff6600, #333333);
}
p{
 padding: 30px 0px 0px 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:

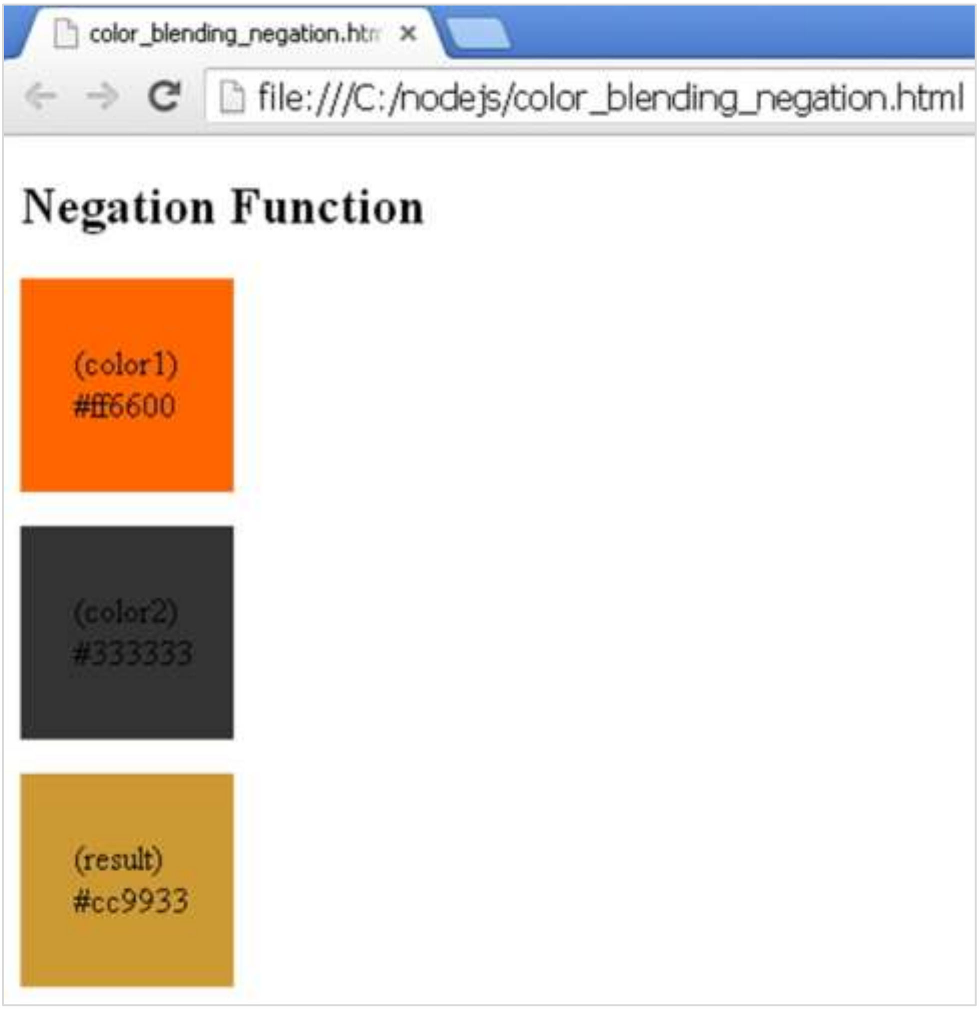
style.css

```
.color1 {
  width: 100px;
  height: 100px;
  background-color: #ff6600;
}
.color2 {
  width: 100px;
  height: 100px;
  background-color: #333333;
}
.result {
  width: 100px;
  height: 100px;
  background-color: #cc9933;
}
p {
  padding: 30px 0px 0px 25px;
}
```

Output

Follow these steps to see how the above code works:

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


color_blending_negation.htm
file:///C:/nodejs/color_blending_negation.html

Negation Function

(color1)
#ff6600

(color2)
#333333

(result)
#cc9933

Usage

34. Less – Command Line Usage

Using the command line, we can compile the *.less* file to *.css*.

Installing lessc for Use Globally

The following command is used to install lessc with npm (node package manager) to make the lessc available globally.

```
npm install less -g
```

You can also add a specific version after the package name. For example, **npm install less@1.6.2 -g**

Installing for Node Development

The following command is used to install the latest version of *lessc* in your project folder.

```
npm i less -save-dev
```

It is also added to the devDependencies in your project package.json.

Beta releases of lessc

It is tagged as beta when the **lessc** structure is published to **npm**. Here, the new functionality is developed periodically. *less -v* is used to get the current version.

Installing an unpublished development version of lessc

The commit - ish is to be specified, when we proceed to install an unpublished version of lessc and the instructions need to be followed for identifying a git URL as a dependency. This will ensure that you are using the correct version of leesc for your project.

Server-Side and Command Line Usage

bin/lessc includes binary in the repository. It works with Windows, OS X and nodejs on *nix.

Command Line Usage

Input is read from **stdin** when source is set to dash or hyphen(-).

```
lessc [option option=parameter ...]  [destination]
```

For instance, we can compile **.less** to **.css** by using the following command:

```
lessc stylesheet.less stylesheet.css
```

We can compile **.less** to **.css** by and minify the result using the following command.

```
lessc -x stylesheet.less stylesheet.css
```

Options

Following table lists out options used in command line usage:

S.NO.	Options & Description	Command

1	**Help**
	Help message is displayed with the options available.
		`lessc -help`
		`lessc -h`

2	**Include Paths** It includes the available paths to the library. These paths can be referenced simply and relatively in the Less files. The paths in windows are separated by colon(:) or semicolon(;).	`lessc --include-path=PATH1;PATH2`

3	**Makefile**
	It generates a makefile import dependencies list to stdout as output.
		`lessc -M`
		`lessc --depends`

4	**No Color** It disables colorized output.	`lessc --no-color`
5	**No IE Compatibility** It disables IE compatibility checks.	`lessc --no-ie-compat`
6	**Disable Javascript** It disables the javascript in less files.	`lessc --no-js`

7	**Lint**
	It checks the syntax and reports error without any output.
		`lessc --lint`
		`lessc -l`

8	**Silent**
	It forcibly stops the display of error messages.
		`lessc --silent`
		`lessc -s`

9	**Strict Imports** It force evaluates imports.	`lessc --strict-imports`
10	**Allow Imports from Insecure HTTPS Hosts** It imports from the insecure HTTPS hosts.	`lessc --insecure`

11	**Version**
	It displays the version number and exits.
		`lessc -version`
		`lessc -v`

12	**Compress**
	It helps in removing the whitespaces and compress the output.
		`lessc -x`
		`lessc --compress`

13	**Source Map Output Filename**
	It generates the sourcemap in less. If sourcemap option is defined without filename then it will use the extension map with the Less file name as source.
		`lessc --source-map`
		`lessc -source-map=file.map`

14	**Source Map Rootpath** Rootpath is specified and should be added to Less file paths inside the sourcemap and also to the map file which is specified in your output css.	`lessc --source-map-rootpath=dev-files/`
15	**Source Map Basepath** A path is specified which has to be removed from the output paths. Basepath is opposite of the rootpath option.	`lessc --source-map-basepath=less-files/`
16	**Source Map Less Inline** All the Less files should be included in the sourcemap.	`lessc --source-map-less-inline`
17	**Source Map Map Inline**	`lessc --source-map-map-inline`

	It specifies that in the output css the map file should be inline.	
18	**Source Map URL** A URL is allowed to override the points in the map file in the css.	`lessc --source-map-url=../my-map.json`

19	**Rootpath**
	It sets paths for URL rewriting in relative imports and urls.
		`lessc -rp=resources/`
		
		`lessc --rootpath=resources/`

20	**Relative URLs**
	
	In imported files, the URL are re-written so that the URL is always relative to the base file.
		`lessc -ru`
		
		`lessc --relative-urls`

21	**Strict Math**
	
	It processes all Math function in your css. By default, it's off.
		`lessc -sm=on`
		
		`lessc --strict-math=on`

22	**Strict Units**
	
	It allows mixed units.
		`lessc -su=on`
		
		`lessc --strict-units=on`

23	**Global Variable** A variable is defined which can be referenced by the file.	`lessc --global-var= "background=green"`
24	**Modify Variable** This is unlike global variable option; it moves the declaration at the end of your less file.	`lessc --modify-var= "background=green"`
25	**URL Arguments** To move on to every URL, an argument is allowed to specify.	`lessc --url-args= "arg736357"`

26	**Line Numbers**
	
	Inline source-mapping is generated.
		`lessc --line-numbers=comments`
		
		`lessc --line-numbers=mediaquery`

		`lessc --line-numbers=all`

27	**Plugin**
	It loads the plugin.
		`lessc --clean-css`
		
		`lessc --plugin=clean-css="advanced"`

35. Less – Using Less In The Browser

Less is used in the browser when you want to compile the Less file dynamically when needed and not on the serverside; this is because less is a large javascript file.

To begin with, we need to add the LESS script to use LESS in the browser:

```
<script src="less.js"></script>
```

To find the style tags on page, we need to add the following line on the page. It also creates the style tags with the compiled css.

```
<link href="styles.less" rel="stylesheet/less" type="text/css"/>
```

Setting Options

Before the script tag, options can be set on the less object programmatically. It will affect all the programmatic usage of less and the initial link tags.

For instance, we can set option as followed:

```
<script>
  less = {
    env: "development"
  };
</script>
<script src="less.js"></script>
```

We can also set the option in another format on the script tag as specified below:

```
<script>
  less = {
    env: "development"
  };
</script>
<script src="less.js" data-env="development"></script>
```

You can also add these options to the link tags.

```
<link data-dump-line-numbers="all" data-global-vars='{ "var": "#fff", "str":
"\"quoted\"" }' rel="stylesheet/less" type="text/css" href="less/style.less">
```

The points that need to be considered for attribute options are:

- *window.less < script tag < link tag* is a level of importance.
- The data attributes cannot be written in camel case; the link tag option are represented as time options.
- The data attributes with non-string values should be JSON valid.

Watch Mode

The watch mode can be enabled by setting the *env* option to *development* and call the *less.watch()* after the less.js file is added. If you want the watch mode to be enabled on a temporary basis, then add *#!watch* to the URL.

```
<script>less = { env: 'development'};</script>

<script src="less.js"></script>

<script>less.watch();</script>
```

Modify Variables

Run time modification of LESS variable is enabled. LESS file is recompiled when new value is called. The following code shows the basic usage of modify variables:

```
less.modifyVars({

  '@buttonFace': '#eee',

  '@buttonText': '#fff'

});
```

Debugging

We can add the option *!dumpLineNumbers:mediaquery* to the url or *dumpLineNumbers* option as mentioned above. The *mediaquery* option can be used with FireLESS(It display the original LESS file name and line number of LESS-generated CSS styles.)

Options

Before loading the script file less.js, options have to be set in a global *less* object.

```
<script>

  less = {

    env: "development",

    logLevel: 2,

    async: false,

    fileAsync: false,

    poll: 1000,
    functions: {},
    dumpLineNumbers: "comments",
    relativeUrls: false,
    globalVars: {
      var1: '"string value"',
      var2: 'regular value'
    },
    rootpath: ":/a.com/"
  };
</script>
<script src="less.js"></script>
```

- **async:** It is a Boolean type. The imported files are requested whether with the option async or not. By default, it is false.
- **dumpLineNumbers:** It is a string type. In the output css file, the source line information is added when the dumpLineNumbers is set. It helps in debugging the particular rule came from.
- **env:** It is a string type. The env may run on development or production. Development is set automatically when the document URL starts with **file://** or it is present in your local machine.
- **errorReporting:** When the compilation fails, the error reporting method can be set.
- **fileAsync:** It is a Boolean type. When a page is present with a file protocol then it can request whether to import asynchronously or not.
- **functions:** It is object type.
- **logLevel:** It is a number type. It displays the logging level in the javascript console.
    - 2: Information and errors
    - 1: Errors
    - 0: Nothing
- **poll:** In the watch mode, the time displays in milliseconds between the polls. It is an integer type; by default, it is set to 1000.
- **relativeUrls:** The URLs adjust to be relative; by default, this option is set as false. This means that the URLs are relative already to the entry less file. It is a Boolean type.
- **globalVars:** Inserts the list of global variables into the code. The string type variable should be included in quotes.
- **modifyVars:** It is unlike the global variable option. It moves the declaration at the end of your less file.

- **rootpath:** It sets paths at the start of every URL resource.
- **useFileCache:** Uses per session file cache. The cache in less files is used to call the modifyVars where all the less files will not retrieve again.

36. Less — Browser support

LESS is cross-browser friendly. It supports modern browsers such as Google Chrome, Mozilla Firefox, Safari and Internet Explorer and allows reusing CSS elements and write LESS code with the same semantics. You must be careful about the performance implications while using LESS on the client side and while displaying the JavaScript to avoid any cosmetic issues such as –

- Spelling mistakes,
- Color changes,
- Texture
- Look
- Links, etc.

Compile the LESS files on the server side to improve the performance levels of the website.

PhantomJS does not implement *Function.prototype.bind* function, so you need to use *es-5 shim* JavaScript engine to run under PhantomJS. Users can make adjustments with variables to affect the theme and show them in real time by using the client side LESS in the production.

If you want to run LESS in older browsers, then use the *es-5 shim* JavaScript engine which adds JavaScript features that LESS supports. You can use attributes on the script or link tags by using *JSON.parse* which must be supported by the browser.

37. Less – Plugins

In this chapter, we will understand how a Plugin can be uploaded to expand the functionality of the site. Plugins can be used to make your work easier.

Command Line

To install plugin using command line, you first need to install the lessc plugin. The plugin can be installed using *less-plugin* at the beginning. The following command line will help you install the clean-css plugin:

```
npm install less-plugin-clean-css
```

Directly, you can use the installed plugin by using the following command:

```
lessc --plugin=path_to_plugin=options
```

Using a Plugin in Code

In Node, the plugin is required and it is pass in an array as an option plugin to the less.

```
var pluginName = require("pluginName");
less.render(myCSS, { plugins: [pluginName] })
   .then(function(output) {
    },
    function(error) {
    });
```

In the Browser

Before the less.js script, the plugin author should include the javascript file in the page.

```
<script src="plugin.js"></script>
<script>
less = {
    plugins: [plugin]
};
</script>
<script src="less.min.js"></script>
```

List of Less Plugins

The following table lists out the plugins available in LESS.

Postprocessor/Feature Plugins

S.NO.	Plugins & Description
1	**Autoprefixer** It is used to add prefixes to CSS after translation from LESS.
2	**CSScomb** It helps to improve the maintenance of your stylesheet.
3	**clean-css** It minifies the CSS output from LESS.
4	**CSSWring** It compresses or minify the CSS output from LESS.
5	**css-flip** It is used to generate the CSS from left-to-right(LTR) or right-to-left(RTL).
6	**functions** It writes the function of LESS in the LESS itself.
7	**glob** It is used to import multiple files.
8	**group-css-media-queries** It does the post-processing for Less.
9	**inline-urls** Automatically converts the URL to data uri.
10	**npm-import** It imports from npm package for less.
11	**pleeease** It is used to postprocess Less.
12	**rtl** LESS is reversed from ltr(left-to-right) to rtl(right-to-left).

Framework/Library Importers

S.NO.	Importers & Description
1	**Bootstrap** Bootstrap LESS code is imported before the custom LESS code.
2	**Bower Resolve** LESS files are imported from the Bower packages.
3	**Cardinal CSS for less.js** Before the custom LESS code, the LESS code for Cardinal is imported.
4	**Flexbox Grid** Most commonly imported Framework or library importer.
5	**Flexible Grid System** It imports the Flexible Grid System.
6	**Ionic** It imports the ionic framework.
7	**Lesshat** It imports the Lesshat mixins.
8	**Skeleton** It imports the skeleton less code.

Function Libraries

S.NO.	Importers & Description
1	**advanced-color-functions** It is used to find more contrasting colors.
2	**cubehelix** Using gamma correction value of 1, the cubehelix function can return a color between the two colors.
3	**lists** This lists manipulation functions library.

For Plugin Authors

LESS allow an author to combine with less.

```
{
    install: function(less, pluginManager) {
    },
    setOptions: function(argumentString) {
    },
    printUsage: function() {
    },
    minVersion: [2, 0, 0]
}
```

- **pluginManager** provides a holder which can add file managers, post processors or visitors.
- **setOptions** function passes the string.
- **printUsage** function is used to explain the options.

38. Less – Programmatic Usage

The main point of programmatic usage in the LESS is *less.render* function. This function uses the following format in LESS:

```
less.render(input_data, options)
    .then(function(output) {
        //code here
    },
    function(error) {
    });
```

the function can also be written in the following way:

```
less.render(css, options, function(error, output) {})
```

The *options* is an optional argument which returns a **promise** when you don't specify the callback and returns a **promise** when you specify the callback. You can display the file by reading it into string and set the filename fields of the main file.

The *sourceMap* option allows to set sourcemap options such as *sourceMapURL*, *sourceMapBasepath*, *sourceMapRootpath*, *outputSourceFiles* and *sour ceMapFileInline*. The point that needs to be considered here is that the *sourceMap* option is not available for the less.js.

You can gain access to the log by adding a listener as shown in the below format:

```
less.logger.addListener({
    debug: function(message) {
    },
    info: function(message) {
    },
    warn: function(message) {
    },
    error: function(message) {
    }
});
```

The above defined functions are optional. If an error is displayed, then it will pass the error to **callback** or **promise** present in the *less.render*.

39. Less – Online Compilers

In this chapter, we will understand the importance of online compilers in LESS. Online compilers are used to compile the less code into css code. Online compilers tools easily help to generate the css code. Following are the available online less compilers:

- less2css.org
- winless.org/online-less-compiler
- lesstester.com
- dopefly.com/less-converte
- lessphp.gpeasy.com/demo
- leafo.net/lessphp/editor
- estFiddle
- ILess

Online Web IDEs/Playgrounds with Less support

Following are the available Online Web IDEs with Less support.

S.NO.	Online Web IDEs & Description
1	**CSSDeck Labs** This is a place where you can easily create testcases that involve HTML, CSS, JS code.
2	**CodePen** This is a playground for the frontend web.
3	**Fiddle Salad** This is a place where you can add an existing code in the environment.
4	**JS Bin** This helps Javascript and CSS code.
5	**jsFiddle** This is an online web editor.

40. Less – GUIs

In this chapter, we will understand the *GUIs for LESS*. You can use different LESS tools for your platform. For *command line usage and tools* click this link.

The following table lists the GUI compilers that supports cross platform.

S.NO.	Tools & Description
1	**Crunch 2!** It supports across platforms like *Windows*, *Mac* and *Linux*. It provides editor with integrated compiling.
2	**Mixture** It is a rapid prototyping and static site generation tool used by designers and developers. It is quick, efficient and works well with your editor. It brings together a collection of awesome tools and best practices.
3	**SimpLESS** It is a minimalistic LESS compiler. It provides drag, drop and compile functionality. SimpLESS supports *prefixing* your CSS by using **prefixr** which is the unique feature of SimpLESS. It is built on Titanium platform.
4	**Koala** It is used to compile LESS, SASS and CoffeeScript. It provides features like compile error notification supports and compile options supports.

The following table lists the GUI compilers that support Windows platform.

S.NO.	Tools & Description
1	**Prepros** It a tool that compiles LESS, SASS, Compass, Stylus, Jade and many more.
2	**WinLess** Initially it was a clone of LESS.app, it has several settings and takes more feature complete approach. It supports starting with command line arguments.

The following table lists the GUI compilers that supports OS X platform.

S.NO.	Tools & Description
1	**CodeKit** It is successor of LESS.app and supports LESS among many other processing languages like SASS, Jade, Markdown and more.
2	**LiveReload** It edits CSS and changes images instantly. SASS, LESS, CoffeeScript and others work well.

The following table lists the GUI compilers that supports OS X platform.

S.NO.	Tools & Description
1	**Plessc** It is gui frontend for lessc. It has features like log viewer, auto compile, opening the LESS file with the chosen editor and sourcemap support.

41. Less – Editors and Plugins

In this chapter, we will understand the importance of *editors and plugins* in LESS. An Editor is a system or program which allows a user to edit text. Plugin is a piece of software that is used to expand the functionality of the site.

Let us now discuss editors and IDEs for LESS.

S.NO.	Editors and IDEs & Description
1	**Crunch!** It supports cross-platforms like *Windows*, *Mac* and *Linux*. It provides editor with integrated compiling.
2	**Mindscape Web Workbench** It provide CoffeeScript, SASS, Compass and LESS editing and makes modern web development easy in Visual Studio.
3	**NetBeans** It is an open-source Java-based IDE. This helps in the quick development of your desktop, mobile and web applications as well as HTML5 applications that involve HTML, JavaScript and CSS.
4	**TextMate** It is a general purpose graphical text editor for Mac OS X. It features declarative customizations, recordable macros, snippets, shell integration, open documents tabs and an extensible bundle system.
5	**Vim** The vim bundle adds functionalities like indenting, highlighting and auto completion for the dynamic stylesheet language LESS.
6	**Emacs** It contains less-css-model that provides an Emacs mode for LESS CSS (lesscss.org); Emacs supports compile-on-save.
7	**jetBrains WebStorm and PhpStorm** WebStrom is a lightweight and powerful IDE. It is perfectly equipped for complex client-side and server development with Node.js. PhpStorm is an PHP IDE, which supports deep code understanding, and provides top-notch coding assistance and support for all major tools and frameworks.
8	**Brackets** It is a lightweight, powerful and an open-source code editor that helps web designers and front-end developers.

9	**CodeLobster** It is a portable integrated development environment (IDE) primarily for PHP. It also supports HTML, CSS and JavaScript development and plugins are available for Drupal, WordPress, Smarty, Joomla, JQuery, Facebook, Codeigniter, Yii and CakePHP.
10	**KineticWing IDE** It is a quick, clean, lightweight and portable IDE. It is a full-size development suite that helps you work smart and fast.
11	**nodeMirror** It is an open-source and easily customizable IDE. It utilizes CodeMirror.net, pty.js and other libraries.
12	**HTML-Kit Tools** This is a modern web editor for HTML5, CSS3, JavaScript and more. With this, you can edit, preview, validate publish and manage projects from modern standards compliant editor.

Sublime Text 2 & 3

The sublime text provides different options for LESS as listed in the following table:

S.NO.	Options & Description
1	**Less-sublime** LESS syntax for sublime text provides syntax highlighting for *.less* files, along with snippets.
2	**Sublime-Less-to-CSS** *Sublime text 2 and 3* plugin to compile *.less* files to CSS when you save. It requires **lessc** installed on PATH.
3	**Less-build-sublime** LESS build system for *sublime text 2* which provides two build systems for LESS files, both minified and non-minified.
4	**SublimeOnSaveBuild** It is a simple plugin for *sublime text 2* to trigger a build when you click Save. Works well with pre-processors like LESS, Compass and any others.

Eclipse

Eclipse has two plugins for LESS as listed in the following table:

S.NO.	Plugins & Description
1	**Eclipse Less Plugin** By extending the Eclipse IDE, this plugin provides useful features to edit and compile LESS stylesheets.
2	**Transpiler Plugin** This Eclipse plugin automatically transpiles your files like LESS, SASS, CoffeeScript, etc.

Visual Studio

Visual Studio has the following different options for LESS:

S.N.	Options & Description
1	**CSS Is Less** This extension makes LESS files open with CSS language service.
2	**Web Essentials 2012** This extension lets you perform common tasks much easier and adds useful features to Visual studio for web developers.
3	**Web Essentials 2013** It extends Visual Studio with a lot of new features which are not specific to a specific language or editor.
4	**Web Tools 2013** This helps you in the development tasks that involve ASP.NET

Dreamweaver

The following points need to be considered while working with Dreamweaver.

- It is an Adobe application used by web designers and developers to create applications and websites.
- It is capable of working across multiple platforms including browsers, devices and tablets
- Web designers use Dreamweaver for creating website prototypes.

- DMXzone Less CSS Compiler makes all the LESS CSS powers directly in Dreamweaver.

Notepad++ 6.x

The following points needs to be considered while working on Notepad++.

- Notepad++ is a free text editor and source code editor which supports tabbed editing, i.e., working with multiple open files in a single window.
- LESS for Notepad++ is an xml file that provides syntax highlighting or coloring for *.less* files. To get more information, click on this link.
- To install Notepad++ click this link.

42. Less – Third Party Compilers

Node.js Compilers

Following are the Node.js compilers used for LESS.

grunt-contrib-less

Grunt is a Node task runner. It will compile your stylesheets every time you commit changes to your LESS files.

assemble-less

assemble-less is a powerful grunt plugin for compiling LESS file to CSS. The less task pulls JSON and Lo - dash(underscore) template for defining the LESS bundles, UI components, compressed stylesheets or themes.

gulp-less

It is LESS plugin for Gulp. *gulp-minify-css* is used to minify your CSS. *gulp-sourcemaps* is used to generate the sourcemaps library.

RECESS

It is an open-source tool which is built on LESS and helps in optimizing your CSS code. It keeps the CSS code error free, clean and manageable.

autoless

It is a *.less* file watcher. It contains dependency tracking and Cross-platform notification.

Connect Middleware for Less.js

It is used to allow the processing for connect JS framework of LESS files. It compiles source file on request and cache the compiled data for next request.

Other Technologies

Following are a few othertechnologies that help you compile a LESS code.

Wro4j Runner CLI

You can download the wro4j-runner.jar and can compile your LESS code in CSS by using the following command:

```
java -jar wro4j-runner-1.5.0-jar-with-dependencies.jar --preProcessors lessCss
```

You can visit the following link to know more about **Wro4j Runner CLI**

CSS::LESSp

This module is used to parse and compile the LESS file into CSS file. Following is the command used to compile:

```
lessp.pl styles.less > styles.css
```

You can visit the following link to know more about **CSS::LESSp**

Windows Script Host

Following is the command line compiler that will run on windows.

```
cscript //nologo lessc.wsf input.less [output.css] [-compress]
```

OR

```
lessc input.less [output.css] [-compress]
```

You can visit the following link to know more about **Less.js for windows**

dotless

The following is a command line compiler to run dotless for windows.

```
dotless.Compiler.exe [-switches]  [outputfile]
```

You can visit the following link to know more about **dotless**

43. Less – Frameworks

UI/Theme Frameworks and Components

LESS supports some of the UI/Theme frameworks as listed in the following table:

S.N.	Framework & Description
1	**1pxdeep** It is flat Bootstrap 3 theme that provides powerful color scheme controls.
2	**Bootflat** It is an open-source framework based on Bootstrap.
3	**BootPress** It is a PHP framework based on flat file CMS.
4	**Bootstrap** It is powerful mobile first front-end framework for faster and easier web development.
5	**Bootstrap a11y theme** It provides easy accessibility for web development.
6	**Bootswatch** It is an open-source theme that provides free themes for Bootstrap.
7	**Cardinal** It is mobile-first CSS framework that allows maintaining CSS for responsive websites, user interfaces, and applications.
8	**CSSHorus** It is a library that provides easy development of mobile websites.
9	**Flat UI Free** It is based on Bootstrap 3 which contains basic and complex components and provides theme design for Bootstrap.
10	**frontsize** It is a frontend framework that contains a set of tools to build widgets.
11	**InContent** It specifies the description of the image using CSS3 and SASS/LESS.

12	**Ink** It creates responsive web interfaces.
13	**JBST** It is powerful theme framework used for creating child themes for WordPress and used as standalone website builder.
14	**KNACSS** It is used to develop HTML/CSS projects by using responsive and extensible style sheets.
15	**Kube** It is CSS framework used for professional designers and developers.
16	**Metro UI CSS** It is a frontend framework used for creating Windows Metro Style on the projects.
17	**Pre** It is CSS framework that uses LESS.
18	**prelude** It is frontend CSS framework that uses LESS.
19	**Schema** It is a light and responsive framework which helps to build complex websites.
20	**Semantic UI** It is a user interface framework that creates responsive layouts using HTML.
21	**UIkit** It is a frontend framework which includes HTML, CSS, and JS components and easy to use and develop web applications.
22	**ngBoilerplate** It is grunt based build system used for AngularJS projects.
23	**less-rail** It is a dynamic stylesheet language that uses Less.js for Rails projects.
24	**Wee** It is a frontend framework which contains HTML, CSS and JavaScript bootstrap components for developing responsive web projects.

Grid Systems

LESS supports grid systems frameworks as listed in the following table:

S.NO.	Framework & Description
1	**Flexible Grid System** It is a CSS framework which creates web projects in a flexible way.
2	**adaptGrid** It is a responsive grid system for developing web applications.
3	**Fluidable** It is lightweight responsive grid system based on LESS preprocessor.
4	**Golden Grid System** It is grid system for responsive design.
5	**LESS Zen Grid** It is used for solving sub pixel rounding issue.
6	**Order.less** It is a LESS library used for alignment, grid system and modular scales.
7	**responsibly** It is a customizable and standalone grid system.
8	**Responsive Boilerplate** It is a lightweight grid system used to create responsive web design for the sites.
9	**Semantic.gs** It is the default distribution of web browser to its operating system.

Mixin Libraries

LESS provides mixin libraries as listed in the following table:

S.NO.	Framework & Description
1	**3L** It provides newest CSS3 features for LESS preprocessor.
2	**animate** It is a library used for browser animations used in the projects.

3	**Clearless** It uses reusable LESS mixins without destroying the style and creating excessive size in stylesheets.
4	**Css3LessPlease** It converts css3please.com to LESS mixins and element will get instant changes when you run the CSS.
5	**CssEffects** It provides CSS style effects written as LESS mixins.
6	**Cssowl** It is a mixin library which supports for LESS, SASS and Stylus.
7	**cube.less** It is a 3D animated cube created using only CSS.
8	**tRRtoolbelt.less** It is a library which provides mixins and functions for performing actions on LESS files.
9	**est** It is based on LESS which allows to write LESS code more efficiently.
10	**Hexagon** It creates CSS hexagons with size and color.
11	**homeless** It is a mixin library that contains helpful functions for the Less.js.
12	**LESS Elements** It is a collection of mixins for the LESS preprocessor.
13	**LESS Hat** It is a mixin library which helps in exporting CSS for all browsers and creates number of shadows, gradients and animations etc.
14	**lessley** It is testing suite which is written in LESS.
15	**LESS-bidi** It is a collection of LESS mixins which provides bi-directional styling without duplication of code.

16	**LESS-Mix** It is a mixin library written in LESS.
17	**media-query-to-type** It is used for creating media queries which allows Internet Explorer 8 and below versions to access the content.
18	**More-Colors.less** It provides variables for color manipulation while designing web applications.
19	**more-less** It is a library which allows to write CSS code for cross browser compatibility.
20	**More.less** It is a combination of Compass and Twitter Bootstrap which provides more to LESS by using CSS3 and cross browser mixins.
21	**more-or-less** It provides powerful mixins for less.js.
22	**normalize.less** It provides normalized CSS using LESS.
23	**Oban** It is a collection of mixins which speeds up the development process of the web application.
24	**Preboot** It is a set of LESS services that uses mixins and variables for writing better CSS and is formed from the Bootstrap.
25	**prelude-mixins** It is a LESS mixin library.
26	**Shape.LESS** It provides a number of mixins for specifying various shapes for the application.