

# Simplifiez-vous la vie avec LESS

Par Iso



[www.openclassrooms.com](http://www.openclassrooms.com)

*Licence Creative Commons 2.2.0  
Dernière mise à jour le 11/10/2011*


## Sommaire

Sommaire .....	2
Lire aussi .....	1
Simplifiez-vous la vie avec LESS .....	3
Installation .....	4
Installation côté client .....	4
Installation côté serveur .....	5
Installation avec Node.js .....	5
Compilation des fichiers pour les servir statiquement .....	5
Utilisation .....	5
Constantes .....	6
Classes abstraites .....	6
Imbrication .....	8
Opérations .....	8
Fonctions .....	9
Commentaires .....	9
Inclusions .....	10
Partager .....	10

## Simplifiez-vous la vie avec LESS



Mise à jour : 11/10/2011

Difficulté : Facile  Durée d'étude : 1 heure, 50 minutes



Le CSS est un langage déclaratif simple : la plupart du temps, on applique une valeur à une propriété. Mais de nos jours, avec le développement des attributs propriétaires (vous savez, les fameux `-moz-*`, `-webkit-*` et autres), le code se trouve dupliqué en de nombreux endroits.

En voici un exemple typique :

Code : CSS

```
#foo {  
  -moz-border-radius: 10px;  
  -webkit-border-radius: 10px;  
  border-radius: 10px;  
  /* etc. */  
}
```

De même, il est impossible d'imbriquer des sélecteurs en CSS :

Code : CSS

```
/* Impossible d'écrire ceci : */  
#foo {  
  bar {  
    color: #150;  
  }  
  baz::before {  
    content: '[';  
  }  
  baz::after {  
    content: ']';  
  }  
}  
  
/* Il vous faudra écrire ceci : */  
#foo bar {  
  color: #150;  
}  
#foo baz::before {  
  content: '[';  
}  
#foo baz::after {  
  content: ']';  
}
```

Enfin, imaginons le cas où il vous faudrait définir une palette de couleurs pour votre site. À chaque changement de couleur, vous devriez éditer vos fichiers à l'aide de la fonction « rechercher / remplacer ». Quelle perte de temps !

Des gens intelligents se sont donc penchés sur la question, et en ont retiré l'observation suivante : « Nous devrions créer un langage *dynamique* capable de générer des feuilles de style que les navigateurs pourraient comprendre. »

Ainsi naquit [LESS](#).

#### Citation : [lesscss.org](#)

LESS extends CSS with dynamic behavior such as variables, mixins, operations and functions. LESS runs on both the client-side (IE 6+, Webkit, Firefox) and server-side, with Node.js.

#### Citation : traduction

LESS étend le CSS avec un comportement dynamique, utilisant des variables, des classes abstraites, des opérations et des fonctions. LESS fonctionne aussi bien côté client (IE version 6 et +, Webkit [Ndt : Safari, Chrome, Midori], Firefox) que côté serveur, avec Node.js.

Il est à noter qu'il existe également des implémentations non officielles en PHP, Ruby, Python, etc.

Commençons dès à présent avec l'installation de LESS.

Sommaire du tutoriel :



- [Installation](#)
- [Utilisation](#)

## Installation

Les navigateurs ne reconnaissent pas LESS, ils ne connaissent que CSS. Il vous faudra donc un interpréteur, qui va transformer LESS en CSS.

Cet interpréteur est capable de s'installer de deux façons différentes :

- côté client (le fichier LESS est traduit par votre visiteur) ;
- côté serveur (vous envoyez directement un fichier CSS).

Il est bien entendu exclu d'installer LESS côté client **et** côté serveur. Vous devez choisir.

### Installation côté client

C'est la méthode la plus simple à mettre en œuvre. Il s'agit d'ajouter un fichier sur votre serveur et de modifier quelque peu la balise `<head>` de votre HTML.

Cependant, cette technique requiert l'activation de JavaScript par le client (le visiteur de votre site). Dans le cas contraire, il n'aura devant les yeux qu'une page non stylisée. C'est la méthode que j'utilise personnellement, étant donné que le nombre de clients n'utilisant pas JavaScript se fait rare et que ceux le désactivant intentionnellement savent à quoi s'en tenir. Aussi, interdiction de se battre dans les commentaires à ce sujet.

En bref, il vous suffira de remplacer la ligne 1 par la ligne 2 :

Code : HTML

```
<!-- Utilise CSS -->
<link rel="stylesheet" type="text/css" href="styles.css">

<!-- Utilise LESS -->
<link rel="stylesheet/less" type="text/css" href="styles.less">
```

Vous devrez télécharger la dernière version de `less.js` depuis [cette liste](#). Je vous conseille d'opter pour une version « production » et « min » (JavaScript minimisé).

Enfin, vous devez rajouter dans votre HTML, **après** vos `<link rel="stylesheet/less" />`, la ligne suivante (c'est important) :

Code : HTML

```
<!-- Changez la source par la position réelle du fichier chez vous
-->
<script src="less.js" type="text/javascript"></script>
```

Vous pouvez maintenant passer à la pratique.

## Installation côté serveur

Si vous décidez d'installer LESS côté serveur, vous pourrez permettre à vos visiteurs de visualiser votre page correctement, même s'ils ont désactivé JavaScript.

Cependant, cette installation est dépendante du serveur utilisé, et il n'existe à ce jour d'interpréteur officiel que pour le fantastique *framework* [Node.js](#).

Deux cas de figure s'offrent maintenant à vous :

- vous utilisez Node.js, et l'installation est décrite plus loin ;
- vous utilisez un autre serveur, et vous souhaitez générer des fichiers CSS que vous servirez statiquement (voir plus loin) ;
- vous utilisez un autre serveur, et devrez vous référer à un moteur de recherche pour l'installation. On se retrouve dans la prochaine partie pour la mise en pratique.

N.B. : si vous voulez rédiger une partie de ce tutoriel pour l'installation de LESS sur un autre logiciel que Node.js, contactez-moi.

## Installation avec Node.js

Vous devez tout d'abord installer le paquet LESS, grâce à *npm* :

**Code : Console**

```
$ npm install less
```

Vous pourrez ensuite utiliser LESS dans votre code en important le module LESS puis en utilisant son compilateur :

**Code : JavaScript**

```
var less = require('less');
less.render(style, callback);
```

## Compilation des fichiers pour les servir statiquement

Vous n'avez qu'à compiler vos fichiers `.less` en `.css` grâce à la commande suivante :

**Code : Console**

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

Vous pourrez ensuite envoyer directement le fichier généré.

Voyons maintenant les avantages de LESS !

## Utilisation

La partie *fun* du cours commence. Vous allez apprendre à utiliser LESS pas à pas. On commence ?

Veuillez noter que CSS est totalement compatible avec LESS, ce qui signifie que vous pourrez tout à fait utiliser du CSS *pur* dans

votre code si vous le souhaitez.

## Constantes

Tout d'abord, LESS vous permet d'utiliser des constantes. Aux férus de programmation, je n'ai rien à apprendre ; pour les autres, retenez simplement que les constantes seront modifiées par leur contenu à la compilation. Un exemple ? En admettant que vous utilisiez les différentes couleurs dans votre longue feuille de style (palette volée sur [colorjack.com](http://colorjack.com)) :

- un bleu électrique (#17B6FF) ;
- un magenta pétant (#FF17B6) ;
- un vert citron (#B6FF17) ;
- un vert acide (#1BFF17).

Si vous décidez de changer une des couleurs de votre palette, vous allez devoir vous battre avec votre éditeur à coups de « rechercher / remplacer ». Ne serait-il pas plus simple de déclarer une constante par couleur ? Avec LESS, vous le pouvez :

**Code : CSS**

```
/* Déclaration */
@blue: #17B6FF;
@magenta: #FF17B6;
@lime: #B6FF17;
@green: #1BFF17;

/* Utilisation */
#header { background: @blue; }
#footer { background: @magenta; }
```

LESS générera le CSS suivant :

**Code : CSS**

```
#header { background: #17B6FF; }
#footer { background: #FF17B6; }
```

On pourrait pousser un peu plus loin le concept en utilisant des noms de variables plus ciblés :

**Code : CSS**

```
/* Déclaration */
@bg_header: #17B6FF;
@bg_footer: #FF17B6;

/* Utilisation */
#header { background: @bg_header; }
#footer { background: @bg_footer; }
```

Vous pouvez utiliser des variables de différents types (couleurs, dimensions, pourcentages, chaînes de caractères, etc.).

## Classes abstraites

Les classes abstraites peuvent être comparées à des fonctions. Expliquons ceci par un exemple (je me permets de reprendre l'exemple d'introduction) :

**Code : CSS**

```
/* Définition de la classe .border-radius */
.border-radius {
  -moz-border-radius: 10px;
  -webkit-border-radius: 10px;
  border-radius: 10px;
}
```

```

}

/* Utilisation de .border-radius comme d'une classe abstraite */
#foo {
  .border-radius;
  /* etc. */
}

/* CSS généré : */
#foo {
  -moz-border-radius: 10px;
  -webkit-border-radius: 10px;
  border-radius: 10px;
  /* etc. */
}

```

Sur un élément, cette transformation peut paraître triviale, mais si vous devez appliquer des bordures arrondies sur 50 éléments, croyez-moi, LESS va vous servir. Cet exemple n'est cependant pas parfait : la valeur 10px est stockée en dur dans notre classe abstraite. Et si nous transmettions des paramètres à notre classe abstraite ?

**Code : CSS**

```

/* Définition de la classe .border-radius avec un paramètre */
.border-radius(@radius) {
  -moz-border-radius: @radius;
  -webkit-border-radius: @radius;
  border-radius: @radius;
}

/* Utilisation de .border-radius comme une classe abstraite */
#foo {
  .border-radius(10px);
  /* etc. */
}

/* CSS généré : */
#foo {
  -moz-border-radius: 10px;
  -webkit-border-radius: 10px;
  border-radius: 10px;
  /* etc. */
}

```

Si vous souhaitez utiliser plusieurs paramètres, séparez-les par des virgules.

Enfin, LESS permet aussi d'utiliser des valeurs par défaut pour les paramètres :

**Code : CSS**

```

/* Définition de la classe .border-radius avec un paramètre
optionnel */
.border-radius(@radius: 10px) {
  -moz-border-radius: @radius;
  -webkit-border-radius: @radius;
  border-radius: @radius;
}

/* Utilisation de .border-radius comme une classe abstraite */
#foo {
  .border-radius(10px);
  /* - ou bien - */
  .border-radius;
  /* etc. */
}

```

```
/* CSS généré : */
#foo {
  -moz-border-radius: 10px;
  -webkit-border-radius: 10px;
  border-radius: 10px;
  /* etc. */
}
```

Il existe une constante dans chaque classe abstraite paramétrable, appelée **@arguments**, et qui contient l'intégralité des paramètres passés à la classe.

Les classes abstraites, comme les constantes, sont très puissantes et vous permettent de minimiser votre feuille de style à moindre coût, et donc de faciliter sa maintenance.

## Imbrication

LESS vous autorise à simplifier votre feuille de style en imbriquant vos sélecteurs, comme vous le feriez avec vos balises HTML. Voici un exemple :

**Code : CSS**

```
/* LESS */
#foo {
  background: lemon;
  bar { color: lightblue; }
  baz { color: orange; }
}

/* CSS généré */
#foo { background: lemon; }
#foo bar { color: lightblue; }
#foo baz { color: orange; }
```

J'imagine que vous n'avez pas besoin d'explications supplémentaires. Un détail cependant : vous pouvez utiliser le symbole **&** comme vous utiliseriez **this** dans un autre langage. Cette comparaison n'est pas très claire, voici donc un exemple :

**Code : CSS**

```
/* LESS */
#foo {
  background: lightgrey;
  &:hover { color: orange; }
}

/* CSS généré */
#foo { background: lightgrey; }
#foo:hover { color: orange; }
```

## Opérations

LESS est capable d'effectuer des opérations. C'est *très* pratique, puisque vous gagnez du temps encore une fois. Exemple :

**Code : CSS**

```
@size_image: 100px;

.box {
  width: @size_image + 10px;
  height: @size_image + 10px;

  img {
```



```
width: @size_image;
height: @size_image;
}
}
```

Dans cet exemple, nous aurions pu utiliser un **margin**, il est vrai. Mais vous n'êtes pas limités aux opérations sur les dimensions !

Code : CSS

```
@bg_body: #555;
@bg_header: @bg_body + #666;

/* @bg_header vaut maintenant #BBB ! */
```

Vous *pouvez* utiliser les parenthèses pour spécifier les priorités dans des opérations complexes, et *devez* les utiliser quand elles sont conjointes à d'autres paramètres :

Code : CSS

```
#header { width: @width - (@space * 2 + @foo); }
#footer { border: (@space / 2) solid orange; }
```

## Fonctions

LESS propose pour l'instant une dizaine de fonctions pour interagir avec les couleurs. Il n'est **pas** possible de définir vos propres fonctions à l'heure actuelle. Ces fonctions sont les suivantes :

Fonction	Description	Exemple
lighten(@c, @x)	Éclaircit	lighten(#AAA, 42%)
darken(@c, @x)	Assombrit	darken(lightblue, 3.14%)
saturate(@c, @x)	Sature	saturate(#123456, 10%)
desaturate(@c, @x)	Désature	desaturate(#567, 50%)
fadein(@c, @x)	Rend transparent	fadein(blue, 30%)
fadeout(@c, @x)	Opacifie	fadeout(rgba(123, 213, 231, 132), 30%)
spin(@c, @x)	Change la teinte de @x degrés	spin(red, 30)
hue(@c)	Retourne la teinte	hue(#345)
saturation(@c)	Retourne la saturation	saturation(lightblue)
lightness(@c)	Retourne la luminosité	lightness(darkgrey)

## Commentaires

LESS vous permet d'écrire vos commentaires à la sauce C++, avec le double *slash* :

Code : C++

```
/* Je suis un commentaire CSS */
// Je suis un commentaire LESS
```

N.B. : j'utilise la coloration C++ dans l'exemple ci-dessus, mais il s'agit bien de LESS.

Il est à noter que si les commentaires CSS sont bien visibles dans vos sources CSS interprétées, ce n'est pas le cas des commentaires LESS. Le CSS généré de l'exemple précédent sera donc :

**Code : CSS**

```
/* Je suis un commentaire CSS */
```

## Inclusions

LESS vous permet de séparer votre code en plusieurs fichiers pour n'en obtenir qu'un au final. C'est l'équivalent d'un `#include` en C. Vous pouvez inclure des fichiers `.less` (l'extension est alors optionnelle) ou `.css`. Attention cependant : dans ce dernier cas, le contenu des fichiers ne sera pas interprété.

**Code : CSS**

```
/* Importation de style LESS */  
@import 'style.less'  
/* - ou bien - */  
@import 'style'  
  
/* Importation de CSS (non interprété) */  
@import 'style.css'
```

Ce n'est pas la fin !

Ce tutoriel vous a montré l'essentiel de LESS, mais il n'est pas complet ! Pour une liste exhaustive des possibilités offertes par ce langage, je vous recommande son site officiel : [lesscss.org](http://lesscss.org).

En attendant, j'espère vous avoir appris comment vous débrouiller avec LESS, et comment produire du CSS de qualité avec un minimum d'efforts !

## Partager



Ce tutoriel a été corrigé par les [zCorrecteurs](#).