

Less, un preprocesador CSS

Adolfo Sanz De Diego

Septiembre 2014

1 El autor

1.1 Adolfo Sanz De Diego

- **Antiguo programador web JEE (6 años)**
- Hoy en día:
 - **Profesor de FP (6 años):**
 - Hardware, Sistemas Operativos
 - Redes, Programación
 - **Formador Freelance (3 años):**
 - Java, Android
 - JavaScript, jQuery
 - JSF, Spring, Hibernate
 - Groovy & Grails

1.2 Algunos proyectos

- Fundador y/o creador:
 - **Hackathon Lovers:** <http://hackathonlovers.com>
 - **Tweets Sentiment:** <http://tweetssentiment.com>
 - **MarkdownSlides:**
<https://github.com/asanzdiego/markdownslides>
- Co-fundador y/o co-creador:
 - **PeliTweets:** <http://pelitweets.com>
 - **Password Manager Generator:**
<http://pasmangen.github.io>

1.3 ¿Dónde encontrarme?

- Mi nick: **asanzdiego**
 - AboutMe: <http://about.me/asanzdiego>
 - GitHub: <http://github.com/asanzdiego>
 - Twitter: <http://twitter.com/asanzdiego>
 - Blog: <http://asanzdiego.blogspot.com.es>
 - LinkedIn: <http://www.linkedin.com/in/asanzdiego>
 - Google+:
<http://plus.google.com/+AdolfoSanzDeDiego>

2 Introducción

2.1 ¿Qué es?

- Less es un **pre-procesador de CSS**.
- Añade características como **variables, mixins, funciones, etc.**

2.2 Ventajas

- El CSS es así **más fácil de mantener, personalizable y extensible.**
- Less (con respecto a otros pre-procesadores CSS) tiene una **sintaxis parecida a CSS.**

2.3 Características

- Less se puede ejecutar **desde NodeJS, desde un navegador, o desde Rhino.**
- Además **existen muchas herramientas** que permiten compilar los archivos y ver los cambios en caliente.

3 Usando Less

3.1 Instalación

- La forma más sencilla de instalar Less, es **a través de la npm**, el gestor de paquetes de NodeJS:

```
$ npm install -g less
```

3.2 Línea de comandos

- Una vez instalado, se puede compilar desde la línea de comandos:

```
$ lessc styles.less > styles.css
```

3.3 Desde NodeJS (I)

- El siguiente código:

```
var less = require('less');  
  
less.render(  
  '.class { width: (1 + 1) }',  
  function (e, css) {  
    console.log(css);  
  }  
);
```

3.4 Desde NodeJS (II)

- Sacará por pantalla:

```
.class {  
  width: 2;  
}
```

3.5 Con Rhino

- **Rhino te permite usar JavaScript desde una Máquina Virtual de Java**

```
java -jar js.jar  
-f less-rhino-<version>.js  
lessc-rhino-<version>.js  
styles.less styles.css
```

3.6 Desde el navegador (I)

- Es la forma más fácil para empezar, pero **no es recomendable usarlo así en producción.**
- **Se recomienda pre-compilar** usando NodeJS, Rhino, o una de las muchas herramientas de terceros disponibles.

3.7 Desde el navegador (II)

- Enlazar tu archivo less que quieras precompilar:

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

3.8 Desde el navegador (II)

- Descargar:
<https://github.com/less/less.js/archive/master.zip>
- Enlazar el js de less:

```
<script src="less.js"  
  type="text/javascript">  
</script>
```

3.9 Desde el navegador (III)

- **Consejos;**
 - Enlaza tus archivos a precompilar antes que la librería de less.
 - Si hay más de un archivo a precompilar, estos se compilan de forma independiente.

4 Características

4.1 Variables (I)

- El siguiente código:

```
@nice-blue: #5B83AD;  
@light-blue: @nice-blue + #111;  
  
#header {  
  color: @light-blue;  
}
```

4.2 Variables (II)

- Les compila a:

```
#header {  
  color: #6c94be;  
}
```

- **Nota: las variables son en realidad "constantes" ya que sólo pueden ser definidas una vez.**

4.3 Extend

- Son una forma de **herencia**:

```
.animal {  
  background-color: black;  
  color: white;  
}  
.bear {  
  &:extend(.animal);  
  background-color: brown;  
}
```

4.4 Mixins (I)

- Los Mixins son una forma de **reutilizar propiedades ya definidas**:
- Imaginemos la clase `.bordered`:

```
.bordered {  
  border-top: dotted 1px black;  
  border-bottom: solid 2px black;  
}
```


4.5 Mixins (II)

- Lo podemos usar así:

```
#menu a {  
  color: #111;  
  .bordered;  
}  
  
.post a {  
  color: red;  
  .bordered;  
}
```

- **Nota: Además de clases, también se pueden utilizar #ids como mixins.**

4.6 Reglas anidadas (I)

- Supongamos que tenemos el siguiente CSS:

```
#header {  
  color: black;  
}  
#header a {  
  color: blue;  
}  
#header a:hover {  
  color: red;  
}
```

4.7 Reglas anidadas (II)

- Pues con Less se puede escribir así:

```
#header {  
  color: black;  
  a {  
    color: blue;  
    &:hover {  
      color: red;  
    }  
  }  
}
```

- **Se pueden usar pseudo-elementos, y llamar al selector padre actual, con &:**

4.8 Medias Queries anidadas (I)

- El siguiente código:

```
@media screen {  
  .screencolor { color: green; }  
}  
@media screen and (min-width: 768px) {  
  .screencolor { color: red; }  
}  
@media tv {  
  .screencolor { color: black; }  
}
```

4.9 Medias Queries anidadas (II)

- Se podría escribir:

```
.screencolor{  
  @media screen {  
    color: green;  
    @media (min-width:768px) {  
      color: red;  
    }  
  }  
  @media tv {  
    color: black;  
  }  
}
```

4.10 Operaciones (I)

- Less puede hacer operaciones con números, colores o variables.
- Además **sabe diferenciar cuando es un número o un color.**

4.11 Operaciones (II)

```
@base: 5%;  
@filler: @base * 2;  
@other: @base + @filler;  
@base-color: #888 / 4;  
  
background-color: @base-color + #111;  
height: 100% / 2 + @other;
```

4.12 Funciones

- Less dispone de una variedad de **funciones matemáticas, que manipulan cadenas, y que transforman los colores:**

```
@base: #f04615;  
@list: 200, 500, 1200;  
  
.class {  
  width: extract(@list, 3);  
  color: saturate(@base, 5%);  
  background-color:  
    lighten(@base, 25%);  
}
```


4.13 Namespaces (I)

- A veces, podemos querer **agrupar mixins**, por motivos de organización, o simplemente para encapsularlos.

4.14 Namespaces (II)

- Veamos como podemos agrupar varios mixins:

```
#bundle {  
  .button {  
    border: 1px solid black;  
    background-color: grey;  
  }  
  .tab { ... }  
  .citation { ... }  
}
```

4.15 Namespaces (III)

- Ahora podemos utilizar el mixin `.button` que está en el namespace `#bundle` de esta forma:

```
#header a {  
  color: orange;  
  #bundle > .button;  
}
```

4.16 Scope

- Los **ámbitos de las variables** en Less es muy similar a otros lenguajes:

```
@var: red;

#page {
  @var: white;
  #header {
    color: @var; // white
  }
}
```

4.17 Comentarios

- Con `//...` y con `/* ... */`

```
/* One hell of a block  
style comment! */  
@var: red;
```

```
// Get in line!  
@var: white;
```

4.18 Imports

```
// foo.less is imported  
@import "foo";
```

```
// foo.less is imported  
@import "foo.less";
```

```
// foo.php imported as a less file  
@import "foo.php";
```

```
// foo.css imported as a css file  
@import "foo.css";
```

5 Variables

5.1 ¿Por qué? (I)

- Las variables se usan **para no tener que repetir constantemente los mismos valores**, con lo que se consigue además un código más fácil de mantener:

```
a,  
.link {  
  color: #428bca;  
}  
.widget {  
  color: #fff;  
  background: #428bca;  
}
```


5.2 ¿Por qué? (II)

- Con Less quedaría:

```
@color: #428bca  
  
a,  
.link {  
  color: @color;  
}  
.widget {  
  color: #fff;  
  background: @color;  
}
```

5.3 Selectores (I)

- También se pueden usar **como selectores**:

```
@mySelector: banner;
```

```
.@{mySelector} {  
  font-weight: bold;  
  line-height: 40px;  
  margin: 0 auto;  
}
```

5.4 Selectores (II)

- Compilado con Less quedaría:

```
.banner {  
  font-weight: bold;  
  line-height: 40px;  
  margin: 0 auto;  
}
```

5.5 URLs

- También se pueden **usar URLs**:

```
@images: "../img";
```

```
body {  
  color: #444;  
  background: url("@{images}/white-sand.png");  
}
```

5.6 Propiedades (I)

- También se pueden usar **como propiedades:**

```
@property: color;  
  
.widget {  
  @property: #0ee;  
  background-@property: #999;  
}
```

5.7 Propiedades (II)

- Compilado con Less quedaría:

```
.widget {  
  color: #0ee;  
  background-color: #999;  
}
```

5.8 Nombres de las variables (I)

- También se pueden usar variables **como nombres de otras variables**:

```
@fnord: "I am fnord.";
@var: "fnord";
content: @@var;
```

5.9 Nombres de las variables (II)

- Compilado con Less quedaría:

```
content: "I am fnord.";
```


5.10 Carga perezosa (I)

- Las variables **no tienen que ser declaradas antes de ser utilizados.**
- Eso es válido:

```
.lazy-eval {  
  width: @var;  
}  
  
@var: @a;  
@a: 9%;
```

5.11 Carga perezosa (II)

- Compilado con Less quedaría:

```
.lazy-eval {  
  width: 9%;  
}
```

5.12 Ámbitos (I)

- Al definir una variable dos veces, **se utiliza la última definición de la variable:**

```
@var: 0;  
.class {  
  @var: 1;  
  .brass {  
    @var: 2;  
    three: @var;  
    @var: 3;  
  }  
  one: @var;  
}
```

5.13 Ámbitos (II)

- Compilado con Less quedaría:

```
.class {  
  one: 1;  
}  
.class .brass {  
  three: 3;  
}
```

6 Extend

6.1 Caso de uso (I)

- Imagino que tenemos lo siguiente:

```
.animal {  
  background-color: black;  
  color: white;  
}
```

- Y queremos tener un subtipo de animal que sobrescriba la propiedad background-color.

6.2 Caso de uso (II)

- Podemos hacer lo siguiente:

```
<a class="animal bear">Bear</a>
```

```
.animal {  
  background-color: black;  
  color: white;  
}  
.bear {  
  background-color: brown;  
}
```

6.3 Caso de uso (III)

- O podemos simplificar el html y usar extend:

```
<a class="bear">Bear</a>
```

```
.animal {  
  background-color: black;  
  color: white;  
}  
.bear {  
  &:extend(.animal);  
  background-color: brown;  
}
```


6.4 Reduce el tamaño del CSS (I)

- Ejemplo de mixin:

```
.my-inline-block() {  
  display: inline-block;  
  font-size: 0;  
}  
.thing1 {  
  .my-inline-block;  
}  
.thing2 {  
  .my-inline-block;  
}
```

6.5 Reduce el tamaño del CSS (II)

- Less lo compila a:

```
.thing1 {  
  display: inline-block;  
  font-size: 0;  
}  
.thing2 {  
  display: inline-block;  
  font-size: 0;  
}
```

6.6 Reduce el tamaño del CSS (III)

- Con extends:

```
.my-inline-block {  
  display: inline-block;  
  font-size: 0;  
}  
.thing1 {  
  &:extend(.my-inline-block);  
}  
.thing2 {  
  &:extend(.my-inline-block);  
}
```

6.7 Reduce el tamaño del CSS (IV)

- Less lo compila a:

```
.my-inline-block,  
.thing1,  
.thing2 {  
  display: inline-block;  
  font-size: 0;  
}
```

7 Mixins

7.1 Selectores

- Se pueden hacer Mixins **tanto con selectores de clase como con selectores de identificación:**

```
.a, #b {  
  color: red;  
}  
.mixin-class {  
  .a();  
}  
.mixin-id {  
  #b();  
}
```

7.2 No exportar Mixins (I)

- Si no quieres que el Mixin sea exportado al CSS, **utiliza los paréntesis:**

```
.my-mixin {  
  color: black;  
}  
.my-other-mixin() {  
  background: white;  
}  
.class {  
  .my-mixin;  
  .my-other-mixin;  
}
```

7.3 No exportar Mixins (II)

- Less lo compila a:

```
.my-mixin {  
  color: black;  
}  
.class {  
  color: black;  
  background: white;  
}
```


7.4 Pseudo-clases (I)

- Los Mixins también soportan **pseudo-clases**:

```
.my-hover-mixin() {  
  &:hover {  
    border: 1px solid red;  
  }  
}  
button {  
  .my-hover-mixin();  
}
```

7.5 Pseudo-classes (II)

- Less lo compila a:

```
button:hover {  
  border: 1px solid red;  
}
```

7.6 Namespaces (I)

- Podemos crear un namespace con varios mixins:

```
#outer {  
  .inner {  
    color: red;  
  }  
}
```

7.7 Namespaces (II)

- Para llamar al Mixin, los parentesis, el espacio y el > es opcional, así que se puede hacer de todas estas formas:

```
#outer > .inner;  
#outer > .inner();  
#outer .inner;  
#outer .inner();  
#outer.inner;  
#outer.inner();
```

7.8 !Important keyword (I)

- Detrás de un Mixin, al compilar pone **todo como importante**:

```
.foo () {  
  background: #f5f5f5;  
  color: #fff;  
}  
.unimportant {  
  .foo();  
}  
.important {  
  .foo() !important;  
}
```

7.9 !Important keyword (II)

- Less lo compila a:

```
.unimportant {  
  background: #f5f5f5;  
  color: #fff;  
}  
.important {  
  background: #f5f5f5 !important;  
  color: #fff !important;  
}
```

8 Mixins paramétricos

8.1 Parámetros (I)

Los Mixins también puede tomar **parámetros**:

```
.border-radius(@radius) {  
  -webkit-border-radius: @radius;  
  -moz-border-radius: @radius;  
  border-radius: @radius;  
}  
  
#header {  
  .border-radius(4px);  
}
```


8.2 Parámetros (II)

- Less lo compila a:

```
#header {  
  -webkit-border-radius: 4px;  
  -moz-border-radius: 4px;  
  border-radius: 4px;  
}
```

8.3 Valor por defecto (I)

- Los Mixins también puede tomar **parámetros con un valor por defecto**:

```
.border-radius(@radius: 5px) {  
  -webkit-border-radius: @radius;  
  -moz-border-radius: @radius;  
  border-radius: @radius;  
}  
  
#header {  
  .border-radius;  
}
```

8.4 Valor por defecto (II)

- Less lo compila a:

```
#header {  
  -webkit-border-radius: 5px;  
  -moz-border-radius: 5px;  
  border-radius: 5px;  
}
```

8.5 Parámetros múltiples (I)

- Los parámetros **se pueden separar por coma (,) o por punto y coma (;).**
- Se recomienda el punto y coma (;).

8.6 Parámetros múltiples (II)

- La coma (,) tiene doble sentido: se puede interpretar como un separador de parámetros Mixin o como separador de los elementos de una lista.
- Si el compilador encuentra al menos un punto y coma (;) asume que los argumentos se separan por punto y coma y los comas pertenecen a listas.

8.7 Parámetros múltiples (III)

- **.name(1, 2, 3; something, else)**
 - 2 parámetros, cada uno es una lista
- **.name(1, 2, 3)**
 - 3 parámetros, cada uno contiene un número
- **.name(1, 2, 3;)**
 - 1 parámetro, que es una lista
- **.name(@param1: red, blue;)**
 - 1 parámetro, con una lista como valor predeterminado

8.8 Parámetros múltiples (IV)

- Se puede tener varios mixins con **el mismo nombre y el mismo número de parámetros**, pues Less utilizará todos los posibles:

```
.mixin(@color) {  
  color: @color;  
}  
.mixin(@color; @padding:2) {  
  padding: @padding;  
}  
.mixin(@color; @padding; @margin: 2) {  
  margin: @margin;  
}  
.some .selector div {  
  .mixin(#008000);  
}
```

8.9 Parámetros múltiples (V)

- Less lo compila a:

```
.some .selector div {  
  color-1: #008000;  
  padding-2: 2;  
}
```


8.10 Parámetros con nombres (I)

- Se pueden usar **parámetros con nombre**:

```
.mixin( @color: black;  
    @margin: 10px;  
    @padding: 20px) {  
    ...  
}  
.class1 {  
    .mixin( @margin: 20px;  
        @color: #33acfe);  
}  
.class2 {  
    .mixin( #efca44;  
        @padding: 40px);  
}
```

8.11 Parámetros con nombres (II)

- Less lo compila a:

```
.class1 {  
  color: #33acfe;  
  margin: 20px;  
  padding: 20px;  
}  
.class2 {  
  color: #efca44;  
  margin: 10px;  
  padding: 40px;  
}
```

8.12 @arguments (I)

- Podemos coger **todos los parámetros de entrada juntos**:

```
.box-shadow(  
  @x: 0;  
  @y: 0;  
  @blur: 1px;  
  @color: #000) {  
  
  -webkit-box-shadow: @arguments;  
  -moz-box-shadow: @arguments;  
  box-shadow: @arguments;  
}  
  
.big-block {  
  .box-shadow(2px; 5px);  
}
```

8.13 @arguments (II)

- Less lo compila a:

```
.big-block {  
  -webkit-box-shadow:  
    2px 5px 1px #000;  
  -moz-box-shadow:  
    2px 5px 1px #000;  
  box-shadow:  
    2px 5px 1px #000;  
}
```

8.14 ...

- Podemos permitir que el Mixin admita **varios parámetros**:

```
// matches 0-N arguments
.mixin(...) {

// matches exactly 0 arguments
.mixin() {

// matches 0-1 arguments
.mixin(@a: 1) {

// matches 0-N arguments
.mixin(@a: 1; ...) {

// matches 1-N arguments
.mixin(@a; ...) {
```

8.15 @rest

- Coge todos los parámetros de ...:

```
.mixin(@a; @rest...) {  
  
  /* @rest recoge todos  
  los parámetros  
  después de @a */  
  
  /* @arguments recoge todos  
  los parámetros  
  (incluido @a) */  
}
```

8.16 Pattern matching (I)

- Si queremos que se ejecute un mixin **dependiendo del valor de una variable:**

```
.mixin(dark; @color) {  
  color: darken(@color, 10%);  
}  
.mixin(light; @color) {  
  color: lighten(@color, 10%);  
}  
.mixin(@_; @color) { /* all */  
  display: block;  
}  
  
@switch: light;  
  
.class {  
  .mixin(@switch; #888);  
}
```

8.17 Pattern matching (II)

- Less lo compila a:

```
.class {  
  color: #a2a2a2;  
  display: block;  
}
```


8.18 Mixins como Funciones (I)

- Todas las variables definidas en un mixin son visibles y **pueden ser utilizados en el ámbito de donde es llamado:**

```
.mixin() {  
  @width: 100%;  
  @height: 200px;  
}  
  
.caller {  
  .mixin();  
  width: @width;  
  height: @height;  
}
```

8.19 Mixins como Funciones (II)

- Less lo compila a:

```
.caller {  
  width: 100%;  
  height: 200px;  
}
```

8.20 Mixins como Funciones (III)

- Otro ejemplo:

```
.average(@x, @y) {  
  @average: ((@x + @y) / 2);  
}  
  
div {  
  
  // "call" the mixin  
  .average(16px, 50px);  
  
  // use its "return" value  
  padding: @average;  
}
```

8.21 Mixins como Funciones (IV)

- Less lo compila a:

```
div {  
  padding: 33px;  
}
```

9 Mixins

Condicionales

9.1 Sintaxis (I)

- Muy parecida a las Media Queries:

```
.mixin (@a)
  when (lightness(@a) >= 50%) {
    background-color: black;
  }
.mixin (@a)
  when (lightness(@a) < 50%) {
    background-color: white;
  }
.mixin (@a) {
  color: @a;
}
.class1 { .mixin(#ddd) }
.class2 { .mixin(#555) }
```

9.2 Sintaxis (II)

- Less lo compila a:

```
.class1 {  
  background-color: black;  
  color: #ddd;  
}  
.class2 {  
  background-color: white;  
  color: #555;  
}
```

9.3 Operadores

- Se pueden usar los operadores `>`, `>=`, `=`, `=<`, `<`

```
@media: mobile;

.mixin (@a)
when (@media = mobile) { ... }

.mixin (@a)
when (@media = desktop) { ... }

.max (@a; @b)
when (@a > @b) { width: @a }

.max (@a; @b)
when (@a < @b) { width: @b }
```


9.4 AND

- Como en las Media Queries, usando **AND** todas las sentencias se tienen que cumplir:

```
.mixin (@a)  
when (isnumber(@a))  
and (@a > 0) { ... }
```

9.5 COMA (,)

- Como en las Media Queries, separar sentencias con **comas (,) equivale a un OR**, por lo que se entrará en el Mixin en cuanto se cumpla una de las sentencias:

```
.mixin (@a)  
  when (@a > 10),  
    (@a < -10) { ... }
```

9.6 NOT

- Como en las Media Queries, usando **NOT** se niega una sentencia:

```
.mixin (@b)  
when not (@b > 0) { ... }
```

9.7 Comprobar tipos

- Tenemos las siguientes funciones para **comprobar tipos**:
 - isnumber
 - isstring
 - iscolor
 - iskeyword
 - isurl

9.8 Comprobar unidades

- Tenemos las siguientes funciones para **comprobar unidades**:
 - ispixel
 - isem
 - ispercentage
 - isunit

9.9 Loops (I)

- Los Mixins se pueden llamar así mismos. Con esta **recursividad** se pueden crear loops:

```
.loop(@counter)
when (@counter > 0) {

  // next iteration
  .loop((@counter - 1));

  // code for each iteration
  width: (10px * @counter);
}

div {
  .loop(5); // launch the loop
}
```

9.10 Loops (II)

- Less lo compila a:

```
div {  
  width: 10px;  
  width: 20px;  
  width: 30px;  
  width: 40px;  
  width: 50px;  
}
```

9.11 Loops (III)

- Podríamos hacer un **grid de CSS**:

```
.generate-columns(4);  
  
.generate-columns(@n, @i: 1)  
  when (@i =< @n) {  
  
    .column-@{i} {  
      width: (@i * 100% / @n);  
    }  
    .generate-columns(@n, (@i + 1));  
  }  
}
```


9.12 Loops (IV)

- Less lo compila a:

```
.column-1 {  
  width: 25%;  
}  
.column-2 {  
  width: 50%;  
}  
.column-3 {  
  width: 75%;  
}  
.column-4 {  
  width: 100%;  
}
```

10 Merge

10.1 ¿Qué es?

- Permite **combinar propiedades** con coma (,) o con espacio (), en una sola propiedad.

10.2 Coma (I)

- Ejemplo con coma (,):

```
.mixin() {  
  box-shadow+: inset 0 0 10px #555;  
}  
.myclass {  
  .mixin();  
  box-shadow+: 0 0 20px black;  
}
```

10.3 Coma (II)

- Less lo compila a:

```
.myclass {  
  box-shadow:  
    inset 0 0 10px #555,  
    0 0 20px black;  
}
```

10.4 Espacio (I)

- Ejemplo con espacio ():

```
.mixin() {  
  transform+_: scale(2);  
}  
.myclass {  
  .mixin();  
  transform+_: rotate(15deg);  
}
```

10.5 Espacio (II)

- Less lo compila a:

```
.myclass {  
  transform: scale(2) rotate(15deg);  
}
```

10.6 Explicito

- Para prevenir cualquier join involuntario, merge **requiere que pongas + o +_ de forma explícita** en la declaración de cada uno de los joins.

11 Selector Padre

11.1 El operador & (I)

- El **operador & representa el selector padre**, y suele ser usado para modificar clases o usar pseudoclases:

```
a {  
  color: blue;  
  &:hover {  
    color: green;  
  }  
}
```

11.2 El operador & (II)

- Less locompila a:

```
a {  
  color: blue;  
}  
  
a:hover {  
  color: green;  
}
```

11.3 Clases repetitivas (I)

- Otro uso, a parte de las pseudoclases, es el de producir nombres de clases repetitivos:

```
.button {  
  &-ok {  
    background-image:  
      url("ok.png");  
  }  
  &-cancel {  
    background-image:  
      url("cancel.png");  
  }  
  &-custom {  
    background-image:  
      url("custom.png");  
  }  
}
```

11.4 Clases repetitivas (II)

- Less lo compila a:

```
.button-ok {  
  background-image: url("ok.png");  
}  
.button-cancel {  
  background-image: url("cancel.png");  
}  
.button-custom {  
  background-image: url("custom.png");  
}
```

11.5 Multiples & (I)

- Se puede repetir el padre:

```
.link {  
  & + & {  
    color: red;  
  }  
  & & {  
    color: green;  
  }  
  && {  
    color: blue;  
  }  
  &, &ish {  
    color: cyan;  
  }  
}
```

11.6 Multiples & (II)

- Less lo compila a :

```
.link + .link {  
  color: red;  
}  
.link .link {  
  color: green;  
}  
.link.link {  
  color: blue;  
}  
.link, .linkish {  
  color: cyan;  
}
```

11.7 Multiples & (III)

- Otro ejemplo:

```
.grand {  
  .parent {  
    & > & {  
      color: red;  
    }  
    & & {  
      color: green;  
    }  
    && {  
      color: blue;  
    }  
    &, &ish {  
      color: cyan;  
    }  
  }  
}
```


11.8 Multiples & (IV)

- Less lo compila a :

```
.grand .parent > .grand .parent {  
  color: red;  
}  
.grand .parent .grand .parent {  
  color: green;  
}  
.grand .parent.grand .parent {  
  color: blue;  
}  
.grand .parent, .grand .parentish {  
  color: cyan;  
}
```

11.9 Cambiar el orden (I)

- En algunos caso puede ser util cambiar el orden del hijo con respecto al padre:

```
.header {  
  .menu {  
    border-radius: 5px;  
    .no-borderradius & {  
      background-image:  
        url('img.png');  
    }  
  }  
}
```

11.10 Cambiar el orden (II)

- Less lo compila a:

```
.header .menu {  
  border-radius: 5px;  
}  
.no-borderradius .header .menu {  
  background-image:  
    url('img.png');  
}
```

11.11 Explosión combinatoria (I)

- El operador & puede ser usado para generar todas las posibles permutaciones de los selectores padre:

```
a, ul, li {  
  border-top: 2px dotted #366;  
  & + & {  
    border-top: 0;  
  }  
}
```

11.12 Explosión combinatoria (II)

- Less lo compila a:

```
a, ul, li {  
  border-top: 2px dotted #366;  
}  
  
a + a, a + ul, a + li,  
ul + a, ul + ul, ul + li,  
li + a, li + ul, li + li {  
  border-top: 0;  
}
```

12 Funciones

12.1 Resumen

- <http://lesscss.org/functions/>
 - Misc Functions
 - String Functions
 - List Functions
 - Math Functions
 - Type Functions
 - Color Functions

12.2 Misc Functions

- <http://lesscss.org/functions/#misc-functions>
 - color
 - convert
 - data-uri
 - default
 - unit
 - get-unit
 - svg-gradient

12.3 String Functions

- <http://lesscss.org/functions/#string-functions>:
 - escape
 - e
 - % format
 - replace

12.4 List Functions

- <http://lesscss.org/functions/#list-functions>
 - length
 - extract

12.5 Math Functions (I)

- <http://lesscss.org/functions/#math-functions>
 - ceil
 - floor
 - percentage
 - round
 - sqrt
 - abs
 - pow
 - mod
 - min
 - max

12.6 Math Functions (II)

- <http://lesscss.org/functions/#math-functions>
 - sin
 - asin
 - cos
 - acos
 - tan
 - atan
 - pi

12.7 Type Functions

- <http://lesscss.org/functions/#type-functions>
 - isnumber
 - isstring
 - iscolor
 - iskeyword
 - isurl
 - ispixel
 - isem
 - ispercentage
 - isunit

12.8 Color Definition Functions

- <http://lesscss.org/functions/#color-definitions>
 - rgb
 - rgba
 - argb
 - hsl
 - hsla
 - hsv
 - hsva

12.9 Color Channel Functions

- <http://lesscss.org/functions/#color-channel>
 - hue
 - saturation
 - lightness
 - hsvhue
 - hsvsaturation
 - hsvvalue
 - red
 - green
 - blue
 - alpha
 - luma
 - luminance

12.10 Color Operation Functions

- <http://lesscss.org/functions/#color-operations>
 - saturate
 - desaturate
 - lighten
 - darken
 - fadein
 - fadeout
 - fade
 - spin
 - mix
 - greyscale
 - contrast

12.11 Color Blending Functions

- <http://lesscss.org/functions/#color-blending>
 - multiply
 - screen
 - overlay
 - softlight
 - hardlight
 - difference
 - exclusion
 - average
 - negation

13 Acerca de

13.1 Licencia

- Estas **transparencias** están hechas con:
 - MarkdownSlides:
<https://github.com/asanzdiego/markdownslides>
- Estas **transparencias** están bajo una licencia Creative Commons Reconocimiento-CompartirIgual 3.0:
 - <http://creativecommons.org/licenses/by-sa/3.0/es>

13.2 Fuentes

- Transparencias:
 - <https://github.com/asanzdiego/curso-interfaces-web-2014/04-less/slides>
- Código:
 - <https://github.com/asanzdiego/curso-interfaces-web-2014/01-less/src>

13.3 Bibliografía

- Documentación oficial de Less
 - <http://lesscss.org>