

# Less, un preprocesador CSS

Adolfo Sanz De Diego  
Septiembre 2016

# 1 El autor

# 1.1 Adolfo Sanz De Diego

- Empecé desarrollando aplicaciones web, hasta que di el salto a la docencia.
- Actualmente soy **Asesor Técnico Docente** en el servicio TIC de la D.G de Infraestructuras y Servicios de la Consejería de Educación, Juventud y Deporte de la Comunidad de Madrid.
- Además colaboro como **formador especializado en tecnologías de desarrollo**.

## 1.2 Algunos proyectos

- **Hackathon Lovers** <http://hackathonlovers.com>: un grupo creado para emprendedores y desarrolladores amantes de los hackathones.
- **Password Manager Generator** <http://pasmangen.github.io>: un gestor de contraseñas online.
- **MarkdownSlides**  
<https://github.com/asanzdiego/markdownslides>: un script para crear slides a partir de ficheros MD.

# 1.3 ¿Donde 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 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 el navegador (I)

- Es la forma más fácil para empezar, pero **no es recomendable usarlo así en producción.**
- Descargar:  
<https://github.com/less/less.js/archive/master.zip>

## 3.4 Desde el navegador (II)

- Enlazar tu archivo less que quieras precompilar, y luego el js de less:

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

# 4 Variables

## 4.1 ¿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;  
}
```

## 4.2 ¿Por qué? (II)

- Con Less quedaría:

```
@color: #428bca

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

## 4.3 Selectores (I)

- También se pueden usar como selectores:

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

## 4.4 Selectores (II)

- Compilado con Less quedaría:

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

# 4.5 URLs

- También se pueden usar URLs:

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

## 4.6 Propiedades (I)

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

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

## 4.7 Propiedades (II)

- Compilado con Less quedaría:

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

## 4.8 Nombres de las variables (I)

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

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

## 4.9 Nombres de las variables (II)

- Compilado con Less quedaría:

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

## 4.10 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%;
```

## 4.11 Carga perezosa (II)

- Compilado con Less quedaría:

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

## 4.12 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;  
}
```

## 4.13 Ámbitos (II)

- Compilado con Less quedaría:

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

# 5 Extend

## 5.1 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.

## 5.2 Caso de uso (II)

- Podemos hacer lo siguiente:

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

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

## 5.3 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;
}
```

## 5.4 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;  
}
```

## 5.5 Reduce el tamaño del CSS (II)

- Less lo compila a:

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

## 5.6 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);  
}
```

## 5.7 Reduce el tamaño del CSS (IV)

- Less lo compila a:

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

# 6 Mixins

## 6.1 ¿Qué son? (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;  
}
```

## 6.2 ¿Qué son? (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.

## 6.3 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();  
}
```

## 6.4 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;  
}
```

## 6.5 No exportar Mixins (II)

- Less lo compila a:

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

# 6.6 Pseudo-clases (I)

- Los Mixins también soportan pseudo-clases:

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

## 6.7 Pseudo-clases (II)

- Less lo compila a:

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

## 6.8 Namespaces (I)

- Podemos crear un namespace con varios mixins:

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

## 6.9 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();
```

## 6.10 !important keyword (!)

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

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

## 6.11 !Important keyword (!)

- Less lo compila a:

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

# 7 Mixins paramétricos

# 7.1 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);  
}
```

## 7.2 Parámetros (II)

- Less lo compila a:

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

## 7.3 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;  
}
```

## 7.4 Valor por defecto (II)

- Less lo compila a:

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

## 7.5 Parámetros múltiples (I)

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

## 7.6 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.

## 7.7 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

## 7.8 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);  
}
```

## 7.9 Parámetros múltiples (V)

- Less lo compila a:

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

## 7.10 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);
}
```

## 7.11 Parámetros con nombres (II)

- Less lo compila a:

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

## 7.12 @arguments ()

- 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);  
}
```

## 7.13 @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;  
}
```

## 7.14 ...

- 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; ...) {
```

## 7.15 @rest

- Coge todos los parámetros de ...:

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

# 7.16 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);  
}
```

## 7.17 Pattern matching (II)

- Less lo compila a:

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

## 7.18 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;  
}
```

## 7.19 Mixins como Funciones (II)

- Less lo compila a:

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

## 7.20 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;  
}
```

## 7.21 Mixins como Funciones (IV)

- Less lo compila a:

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

# 8 Mixins Condicionales

# 8.1 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) }
```

## 8.2 Sintaxis (II)

- Less lo compila a:

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

## 8.3 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 }
```

## 8.4 AND

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

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

## 8.5 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) { ... }
```

## 8.6 NOT

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

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

## 8.7 Comprobar tipos

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

## 8.8 Comprobar unidades

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

## 8.9 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
}
```

## 8.10 Loops (II)

- Less lo compila a:

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

# 8.11 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));  
}
```

## 8.12 Loops (IV)

- Less lo compila a:

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

# 9 Merge

## 9.1 ¿Qué es?

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

## 9.2 Coma (l)

- Ejemplo con coma (,):

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

## 9.3 Coma (II)

- Less lo compila a:

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

## 9.4 Espacio (I)

- Ejemplo con espacio ():

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

## 9.5 Espacio (II)

- Less lo compila a:

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

## 9.6 Explicito

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

# 10 Selector Padre

# 10.1 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;  
  }  
}
```

## 10.2 El operador & (||)

- Less lo compila a:

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

## 10.3 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");  
  }  
}
```

## 10.4 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");  
}
```

# 10.5 Multiples & ()

- Se puede repetir el parentesis:

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

# 10.6 Multiples & (II)

- Less lo compila a :

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

# 10.7 Multiples & (III)

- Otro ejemplo:

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

# 10.8 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;  
}
```

## 10.9 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');  
    }  
  }  
}
```

## 10.10 Cambiar el orden (II)

- Less lo compila a:

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

## 10.11 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;  
    }  
}
```

## 10.12 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;  
}
```

# 11 Funciones

## 11.1 Resumen (I)

- 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%);  
}
```

## 11.2 Resumen (II)

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

# 11.3 Misc Functions

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

## 11.4 String Functions

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

## 11.5 List Functions

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

# 11.6 Math Functions (I)

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

# 11.7 Math Functions (II)

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

# 11.8 Type Functions

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

# 11.9 Color Definition Functions

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

# 11.10 Color Channel Functions

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

# 11.11 Color Operation Functions

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

# 11.12 Color Blending Functions

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

# 12 Acerca de

## 12.1 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>

## 12.2 Fuentes

- Transparencias:
  - <https://github.com/asanzdiego/curso-interfaces-web-2016/tree/master/04-less/slides>
- Código:
  - <https://github.com/asanzdiego/curso-interfaces-web-2016/tree/master/01-less/src>

## 12.3 Bibliografía

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