
CSS3

CSS3

La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño. Debido a esta consideración, la **integración** entre **HTML** y **CSS** es ahora vital para el desarrollo web y esta es la razón por la que cada vez que mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas.

Las nuevas incorporaciones de CSS3 se van implementando en las últimas versiones de los navegadores más populares, pero algunas de ellas se encuentran aún en estado experimental. Por esta razón, **algunos de estos nuevos estilos deberán ser precedidos por prefijos** para ser efectivamente interpretados.

-moz-	*para Firefox.*
-webkit-	*para Safari y Chrome.*
-chrome-	*solo para Chrome.*
-ms-	*para Internet Explorer.*
-o-	*para Opera.*
-khtml-	*para Konqueror.*

Referenciando con cualquier atributo

Los métodos de referencia estudiados para CSS cubren un variado espectro de situaciones, pero a veces no son suficientes para encontrar el elemento exacto dentro del documento. La última versión de CSS ha incorporado nuevas formas de referenciar elementos HTML.

Uno de ellos es el **Selector de Atributo**. Ahora podemos referenciar un elemento no solo por los atributos *id y class* sino también a través de **cualquier otro atributo**:

p[name="mitexto"] { font-size: 20px }

CSS3 permite combinar "=" con *otros símbolos* para hacer una selección más específica:

```
p[name^="mi"] {  font-size: 20px  }
p[name$="mi"] {  font-size: 20px  }
p[name*="mi"] {  font-size: 20px  }
```

- La regla con el selector **^=** será asignada a todo elemento <p> que contiene un atributo **name** con un valor **comenzado en "mi"** (por ejemplo, "mitexto", "micasa").

- La regla con el selector **$=** será asignada a todo elemento <p> que contiene un atributo **name** con un valor **finalizado en "mi"** (por ejemplo "textomi", "casami").

- La regla con el selector ***=** será asignada a todo elemento <p> que contiene un atributo **name** con un valor que **incluye el texto "mi"** (en este caso, el texto podría también encontrarse en el medio, como en "textomicasa").

Referenciando con pseudoclases

CSS3 también incorpora **nuevas pseudoclases** que hacen la selección aún más específica.

Imaginemos un documento HTML con un que tiene cuatro elementos <p> en su interior, que son hermanos porque están al mismo nivel. Esto significa que todos ellos tienen el mismo padre que es el elemento .

```
p:nth-child(2){
    background: #999999;
}
```

La regla **p:nth-child(2)** referencia a cada segundo elemento <p> encontrado en el documento.

La *pseudoclase* es agregada usando **dos puntos antes del su nombre**. Esta regla puede incluir otras referencias. Por ejemplo, podríamos escribirla como `.miclase:nth-child(2)` para referenciar todo elemento que es hijo de otro elemento con *class* igual a *miclase*.

Una alternativa, que genera menos código y resulta interesante, es aprovechar las palabras clave **odd** (impares) y **even** (pares) disponibles para esta *pseudoclase*:

```
p:nth-child(odd){
    background: #999999;
}
p:nth-child(even){
    background: #CCCCCC;
}
```

Existen otras pseudoclases relacionadas con esta última, como **firstchild, last-child** y **only-child**:

- **first-child** referencia solo el primer hijo
- **last-child** referencia solo el último hijo
- **only-child** afecta un elemento siempre y cuando sea el único hijo disponible.

Otra importante *pseudoclase* llamada **not()** es utilizada realizar una negación. Los estilos en la regla creada con esta pseudoclase serán asignados a todo elemento excepto aquellos incluidos en la referencia entre paréntesis.

```
:not(p){
margin: 0px;
}
```

Esta regla del asignará un margen de 0 píxeles a cada elemento del documento **excepto los elementos <p>**.

En lugar de la palabra clave de un elemento podemos usar cualquier otra referencia que deseemos.

Nuevos selectores

Estos nuevos selectores usan los símbolos **>**, **+** y **~** para especificar la relación entre elementos.

El selector > está indicando que el elemento a ser afectado por la regla es el elemento de la derecha cuando tiene al de la izquierda como su padre.

```
div > p.mitexto2 {
    color: #990000;
}
```

La regla mostrada arriba modifica los **elementos <p> que son hijos de un elemento **. En este caso, referenciamos solamente el elemento <p> con el valor *mitexto2* en su atributo *class*.

El selector + referencia al elemento de la derecha cuando es inmediatamente precedido por el de la izquierda. Ambos elementos deben compartir el mismo padre.

```
p.mitexto2 + p {
    color: #990000;
}
```

La regla anterior afecta al elemento <p> que se encuentra ubicado después de otro elemento <p> identificado con el valor mitexto2 en su atributo class.

El selector ~ es similar al anterior pero el elemento afectado no necesita estar precediendo de inmediato al elemento de la izquierda. Además, más de un elemento puede ser afectado:

```
p.mitexto2 ~ p {
    color: #990000;
}
```

El estilo será aplicado a todos los elementos <p> que son hermanos y se encuentran después del elemento <p> identificado con el valor mitexto2 en su atributo class. **No importa si otros elementos se encuentran intercalados.**

Más selectores

Pseudo-clases

:enabled

Un elemento de interfaz de usuario que está **habilitado**.

:disabled

Por contra, un elemento de interfaz de usuario que está **deshabilitado**.

:checked

Los botones **de radio** o casillas de verificación (**checkboxes**) que están **seleccionados**

:target

Selecciona el elemento que es el objetivo **del ancla activa actualmente** de dentro de la página. Recordamos a que se pueden tener enlaces en anclas dentro de una página **utilizando el carácter # con el id del objetivo.** Por ejemplo, `<a href="#content"> Ir al Contenido </a>`

:default

Se aplica a uno o más elementos de interfaz de usuario que tienen el **valor por defecto**.

:valid

Elementos **válidos**, de acuerdo con el atributo ***pattern*** (patrón de expresión regular).

:invalid

Se aplica a elementos definidos como ***required*** que están vacíos o a elementos que no siguen los requisitos del tipo o del atributo ***pattern***.

:in-range

Se aplica a los elementos con limitaciones ***de rango***, dentro de las limitaciones.

:out-of-range

Elementos cuyo valor está fuera de los límites de su rango definido.

:required

Selecciona controles **de formulario** que tienen activado el atributo ***required***.

:optional

Se aplica a los controles **de formulario** que no tienen el atributo ***required*** activado.

:read-only

Se aplica a los elementos cuyos **contenidos no pueden ser modificados por el usuario.** Esto es, en general, la mayoría de los elementos que no son campos de formulario.

:read-write

Se aplica a elementos cuyo contenido es **modificable por el usuario.**

Pseudo-clases estructurales

Hemos visto antes como podemos apuntar a los elementos en función de sus atributos y estados.

También es posible seleccionar elementos basados en su **ubicación en el código** de marcado.

:root

El elemento **raíz**, que es siempre el elemento **html**.

E:nth-of-type(n)

El elemento que es el elemento enésimo **de su tipo en un elemento padre** (E) dado.

E:nth-last-of-type(n)

Como el anterior, excepto que ahora se cuenta **hacia atrás** desde el último elemento.

E:empty

Un elemento **que no tiene hijos**.

Pseudo-elementos y contenido generado

Los **pseudoelementos** representan algún tipo de estructura del documento que está fuera del DOM. Por ejemplo, todos los nodos de texto tienen una **primera letra** y una **primera línea**, pero ¿cómo se pueden seleccionar explícitamente sin meterlos dentro de un span?

CSS proporciona los pseudoelementos **::first-letter** y **::first-line** que coinciden con la primera letra y la primera línea de un nodo de texto, respectivamente.

::after

Permite **insertar contenido** en una página desde CSS (sin que necesite estar en HTML). Aunque el resultado final **no aparece en el DOM**, aparece en la página como si lo estuviera.

```
a[href$=pdf] {
    background: transparent url('icon.gif') 0 50% no-repeat;
    padding-left: 20px;
}
a[href$=pdf]:after {
    content: "(PDF)";
}
```

Estos estilos añadirán un icono de pdf y el texto "(PDF)" después de enlaces a archivos pdf.

::before

Lo mismo que **::after**, solo que inserta **el contenido antes** de cualquier otro contenido.

::selection

Se aplica al **texto** que está **resaltado**.

```
::-moz-selection{
    background:#484848;
    color:#fff;
}
::selection{
    background:#484848;
    color:#fff;
}
```

Nuevas reglas

border-radius

Esta propiedad genera **esquinas redondeadas** para la caja formada por el elemento. Posee dos parámetros diferentes que dan forma a la esquina. El primer parámetro determina la curvatura horizontal y el segundo la vertical, otorgando la posibilidad de crear una elipse.

Para declarar ambos parámetros de la curva, los valores deben ser separados por una barra (por ejemplo, **border-radius: 15px / 20px**).

```
-moz-border-radius: 20px / 10px;
-webkit-border-radius: 20px / 10px;
border-radius: 20px / 10px;
```

Usar un valor determina la misma forma para todas las esquinas (e.g. `border-radius: 20px`). Un valor para cada esquina puede ser declarado, en orden de las agujas del reloj.

```
-moz-border-radius: 20px 10px 30px 50px;
-webkit-border-radius: 20px 10px 30px 50px
border-radius: 20px 10px 30px 50px;
```

box-shadow

Esta propiedad crea sombras para la caja formada por el elemento. Puede tomar 5 parámetros:

- El **color**
- El **desplazamiento** horizontal (puede ser positivo o negativo)
- El **desplazamiento** vertical (puede ser positivo o negativo)
- El valor de **difuminación** (opcional)
- La palabra clave **inset** para generar una sombra interna (opcional)

Por ejemplo, `box-shadow: #000000 5px 5px 10px inset;`

Los valores de los desplazamientos indican, respectivamente, la distancia horizontal y vertical desde la sombra al elemento. Valores negativos posicionarán la sombra a la izquierda y arriba del elemento, mientras que valores positivos crearán la sombra a la derecha y debajo del elemento.

```
-moz-box-shadow: rgb(150,150,150) 5px 5px;
-webkit-box-shadow: rgb(150,150,150) 5px 5px;
box-shadow: rgb(150,150,150) 5px 5px;
```

text-shadow

Esta propiedad es similar a *box-shadow* pero específica **para textos**. Toma 4 parámetros:

- El **color**
- El desplazamiento **horizontal**
- El desplazamiento **vertical**
- El valor de **difuminación**

Por ejemplo, `text-shadow: #000000 5px 5px 10px`.

@font-face

Esta regla nos permite **cargar y usar** cualquier fuente que necesitemos. Primero, debemos **declarar la fuente**, proveer un nombre con la propiedad font-family y especificar el archivo:

Por ejemplo,

```
@font-face{
    font-family: MiNuevaFuente;
    src: url('font.ttf');
}
```

Después, podremos asignar la fuente a cualquier elemento del documento.

```
#titulo {
    font: bold 36px MiNuevaFuente, verdana, sans-serif;
    text-shadow: rgb(0,0,150) 3px 3px 5px;
}
```

linear-gradient

Esta función puede ser aplicada a las propiedades **background** o **background-image** para generar un gradiente lineal. Los atributos indican:

- El **punto** inicial.
- **Color inicial** del gradiente.
- **Color final** del gradiente.

El primer valor puede ser especificado:

- En **píxeles**
- En **porcentaje**
- Usando las **palabras clave**: *top, bottom, left y right.*

Por ejemplo, `background: linear-gradient(top, #FFFFFF 50%, #006699 90%);`

Para la **dirección**, puede proporcionarse ya sea el ángulo por el que el gradiente debe proceder, o el **lado o esquina** donde debe empezar. En este último caso se procederá hacia el lado o esquina opuesto.

Para los **ángulos**, se utiliza el valor en grados (°). Usaremos **0deg** para apuntar hacia la derecha, y **90deg** apunta hacia arriba y **así sucesivamente en sentido contrario a las agujas del reloj**. Para un lado o esquina, se utilizan las palabras clave **top, bottom, left, y right.**

Después de especificar la dirección, hay que proporcionar los ***color-stops***; estos **se componen de un color y un porcentaje** o longitud que especifica hasta qué punto a lo largo del gradiente llega este color-stop. Por ejemplo, para ir de negro a blanco, y después de nuevo a negro:

```
background-image: -moz-linear-gradient(30deg, #000, #fff 75%, #000 90%);
```

Si se omiten los color-stops se distribuirán de manera uniforme.

radial-gradient

Esta función puede ser aplicada a las propiedades **background** o **background-image** para generar un gradiente radial. Los atributos son: ***posición inicio, forma, color inicial, color final***

La **posición de inicio** es el origen y puede ser declarado en **píxeles,** en **porcentaje**, o como una combinación de las **palabras clave** center, top, bottom, left y right.

Existen dos valores para la **forma**: **circle** o **ellipse**

La sintaxis de ***color-stops*** es la misma que por *gradientes lineales*: un valor de color seguido de una posición de parada opcional.

```
background: -moz-radial-gradient(center, circle, #FFFFFF 10%, #006699 90%);
background: -webkit-radial-gradient(center, circle, #FFFFFF 10%, #006699 90%);
```

Como **mínimo**, hay que indicar un **color de inicio** y un **color final**. Opcionalmente, también se puede indicar una posición para **el centro** del gradiente, y una **forma**.

```
background-image: -moz-radial-gradient(#fff, #000);
background-image: -moz-radial-gradient(top, #fff, #000);
background-image: -moz-radial-gradient(top, circle, #fff, #000);
```

rgba()

Esta función es una **mejora de rgb()**. Toma cuatro valores:

- El color **rojo** (0-255).
- El color **verde** (0-255).
- El color **azul** (0-255).
- La **opacidad** (un valor entre 0 y 1).

```
text-shadow: rgba(0,0,0,0.5) 3px 3px 5px;
```

hsla()

Esta función es una **mejora de hsl()**. Puede tomar cuatro valores:

- El **tono** (un valor entre 0 y 360).
- La **saturación** (un porcentaje).
- La **luminosidad** (un porcentaje).
- La **opacidad** (un valor entre 0 y 1).

```
color: hsla(120, 100%, 50%, 0.5);
```

outline

Esta propiedad fue mejorada con la incorporación de otra propiedad llamada outline-offset. Ambas propiedades combinadas generan un **segundo borde alejado del borde original del elemento**.

Por ejemplo,

```
#principal {
    display: block;
    width: 500px;
    margin: 50px auto;
    padding: 15px;
    text-align: center;
    border: 1px solid #fd2525;
    background: rgb(248, 245, 200);
    outline: 2px dashed #81b1ff;
    outline-offset: 5px;
}
```

border-image

Crea un **borde con una imagen personalizada**. Necesita que el borde sea **declarado previamente** con las propiedades **border** o **border-width**, y toma al menos tres parámetros:

- La **URL** de la imagen
- El **tamaño** de las piezas que serán tomadas de la imagen para construir el borde
- Una ***palabra clave*** que especifica cómo esas piezas serán ubicadas alrededor del elemento
  - ***repeat*** repetirá las piezas tomadas de la imagen todas las veces que sea necesario para cubrir el lado del elemento. En este caso, el tamaño de las piezas es preservado y la imagen será cortada si no existe más espacio para ubicarla.
  - ***round*** considerará cómo de largo es el lado a ser cubierto y ajustará el tamaño.
  - ***stretch*** estira solo una pieza para cubrir el lado completo.

Examples:

https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_border-image2

https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_border-image

https://www.w3schools.com/cssref/playit.asp?filename=playcss_border-image2

transform

Esta propiedad **modifica la forma de un elemento**. Utiliza cuatro funciones básicas:

- **scale** (escalar).
- **rotate** (rotar).
- **skew** (inclinar).
- **translate** (trasladar o mover).

transform: scale

Si recibe dos parámetros: el valor X para la escala horizontal y el valor Y para la escala vertical. Si solo un valor es provisto, el mismo valor es aplicado a ambos parámetros.

- un valor **negativo invierte** el elemento
- valores **entre 0 y 1 reducen** el elemento
- valores **mayores que 1 expanden** el elemento

Por ejemplo,

```
-moz-transform: scale(1.5,-1.5);
-webkit-transform: scale(1.5,-1.5);
```

transform: rotate

Usa solo un parámetro expresado en grados para rotar el elemento.

```
-moz-transform: rotate(30deg);
-webkit-transform: rotate(30deg);
```

transform: skew

La función skew (inclinar) recibe dos valores, también en grados, para la **transformación horizontal y vertical**. Si solo declaramos el primer parámetro, solo la dimensión horizontal de la caja será modificada.

```
-moz-transform: skew(20deg, 10deg);
-webkit-transform: skew(20deg, 10deg);
```

transform: translate

Mueve el objeto tantos pixeles como sean especificados por sus parámetros. Dos valores pueden ser declarados en esta función para movimiento horizontal y vertical, o podemos usar funciones independientes llamadas **translateX** y **translateY**.

La **esquina superior izquierda** del elemento es la **posición 0,0,** por lo que:

- Valores **negativos** mueven el objeto hacia la **izquierda** o **arriba** de la posición original.
- Valores **positivos** lo harán hacia la **derecha** o hacia **abajo**.

Transformando todo al mismo tiempo

Para realizar sobre un elemento varias transformaciones al mismo tiempo, tenemos que separar cada función a aplicar con un espacio:

```
-moz-transform: translateY(100px) rotate(45deg) scaleX(0.3);
-webkit-transform: translateY(100px) rotate(45deg) scaleX(0.3);
```

Transformaciones dinámicas

Podemos aprovecharnos de la **combinación de transformaciones y pseudoclases** para convertir nuestra página en una aplicación dinámica.

```
#principal:hover{
    -moz-transform: rotate(5deg);
    -webkit-transform: rotate(5deg);
}
```

transition

Esta propiedad puede ser aplicada para crear una **transición entre dos estados de un elemento**. Recibe hasta cuatro parámetros:

- La **propiedad** afectada
- El **tiempo** que le tomará a la transición desde el comienzo hasta el final
- Una **palabra clave** para especificar cómo la transición será realizada (**ease**, **linear**, **ease-in**, **ease-out**, **ease-in-out**)
- Un valor de **retardo** que determina el tiempo que la transición tardará en comenzar.

Por ejemplo, `transition: color 2s linear 1s;` *(versión abreviada)*

```
#principal {
    background-color: lightgreen;
    -moz-transition: background-color 1s ease-in-out 0.1s;
    -webkit-transition: background-color 1s ease-in-out 0.1s;
    transition: background-color 1s ease-in-out 0.1s;
}

#principal:hover{
    background-color: lightsalmon;
}
```

Vamos a aplicar la transición a la propiedad **transform**, por ejemplo:

```
#info {
    -webkit-transition-property: -webkit-transform;
    -moz-transition-property: -moz-transform;
    -o-transition-property: -o-transform;
    transition-property: transform;
}
```

Hasta ahora estos estilos no tendrán efecto. Debemos **especificar la duración de la transición**.

La propiedad **transition-duration** fija cuánto tiempo durará la transición. Se puede especificar en segundos (**s**) o milisegundos (**ms**). Por ejemplo, 0,2 segundos, o 200 milisegundos,

```
    -webkit-transition-duration: 0.2s;
    -moz-transition-duration: 0.2s;
    -o-transition-duration: 0.2s;
    transition-duration: 0.2s;
```

Con **transition-timing-function** se permite controlar el ***ritmo de la transición***. Podremos indicar si queremos que una animación empiece lenta y acabe más rápida, empiece rápida y acabe más lenta, avance a un nivel estable, o alguna otra variación. Se puede especificar uno de los valores clave **ease, lineal, ease-in, ease-out, o ease-in-out**.

```
-webkit-transition-timing-function: ease-out;
-moz-transition-timing-function: ease-out;
-o-transition-timing-function: ease-out;
transition-timing-function: ease-out;
```

Mediante el uso de la propiedad **transition-delay**, también es posible introducir un retraso antes de que empiece la animación. Se puede incluir el número de milisegundos (**ms**) o segundos (**s**) por atrasar la transición:

```
-webkit-transition-delay: 250ms;
-moz-transition-delay: 250ms;
-o-transition-delay: 250ms;
transition-delay: 250ms;
```

Se puede proporcionar cualquier número de propiedades CSS a la declaración de propiedad de **transition-property**, separadas por comas. Alternadamente, se puede utilizar la palabra llave **all** por indicar que toda propiedad debe ser animada.

También se pueden especificar diferentes duraciones y funciones de temporización para cada propiedad que está siendo animada. Solo se debe incluir cada valor en una lista **separada por comas,** utilizando el mismo orden que en **transition-property**:

```
transition-property: transform, color;
transition-duration: 0.2s, 0.1s;
transition-timing-function: ease-out, lineal;
```

Se pueden especificar también múltiples transiciones cuando se utiliza la versión **abreviada** de la propiedad **transition**. En este caso, se deben especificar todos los valores para cada transición juntos, y separar cada transición con comas:

```
transition: color 0.2s ease-out, transform 0.2s ease-out;
```

Usando la palabra clave **all**, todas las propiedades cambian al mismo ritmo, velocidad, y retraso:

```
-webkit-transition: all 0.2s ease-out;
-moz-transition: all 0.2s ease-out;
-o-transition: all 0.2s ease-out;
transition: all 0.2s ease-out;
```

Animaciones

Las animaciones CSS, a diferencia de las transiciones, permiten controlar cada paso de una animación mediante fotogramas clave. Para crear una animación, se utiliza la regla **@keyframes** seguida por un nombre descriptivo, que servirá como el identificador para la animación. A continuación, se pueden especificar sus fotogramas clave.

En lugar de un selector, se utilizan las palabras clave ***from*** o ***to***, un valor porcentual o una lista de valores porcentuales separada por comas.

```
@keyframes appearDisappear {
    0%, 100% {
        opacity: 1;
    }
    20%, 80% {
        opacity: 0;
    }
}
```

Una vez que se ha definido una animación, el siguiente paso es aplicarla a uno o más elementos usando las diversas propiedades de animación.

Propiedades de animación

animation-name

Esta propiedad se utiliza para **adjuntar** una **animación** a un **elemento**.

```
animation-name: appearDisappear;
```

animation-duration

Define el período de tiempo, en segundos o milisegundos, que una animación tarda en completar una iteración (desde 0% a 100%):

```
animation-duration: 300ms;
```

animation-timing-function

Al igual que la propiedad de transition-timing-function, determina la forma en la que la animación progresará sobre su duración.
Las opciones son las mismas: *ease, linear, ease-in, ease-out, ease-in-out, o cubic-bezier:*

```
animation-timing-function: linear;
```

animation-iteration-count

Esta propiedad permite definir la cantidad de veces que la animación se reproducirá. También se pueden usar números con puntos decimales (en cuyo caso, la animación va a terminar a mitad de la reproducción), o el valor **`infinite`** para repetir sin cesar.

```
animation-iteration-count: infinite;
```

animation-direction

Cuando la animación se repite, se puede utilizar la propiedad de ***`animationdirection`*** con el valor **`alternate`** para reproducir la animación hacia delante y hacia atrás.

```
animation-direction: alternate;
```

Por ejemplo, en una animación de pelota que rebota, puede proporcionar fotogramas clave para la pelota cayendo, y luego usar `alternate` para invertirla en cada segunda reproducción.

animation-fill-mode

La propiedad **`animation-fill-mode`** define lo que sucede antes de que la animación comience y después de que la animación termine.

Por defecto, una animación no afectará al valor de las propiedades fuera de sus reproducciones, pero con **`animation-fill-mode`**, podemos anular este comportamiento predeterminado. Le indicamos a la animación que "espere" en el primer fotograma clave hasta que se inicie la animación, o que pare en el último fotograma clave sin volver a los valores originales en la conclusión de la animación, o ambos.

Los valores disponibles son **`none`**, **`forwards`**, **`backwards`**, **o** **`both`**.

- El valor predeterminado es **`none`**. Vuelve al fotograma clave inicial al terminar.
- Cuando se establece el valor **`forwards`**, continúa aplicando los valores de los últimos fotogramas clave después del final de la animación.
- Cuando se ponen **`backwards`**, los fotogramas clave iniciales de la animación se aplican tan pronto como el estilo de animación se aplica a un elemento.
- Como era de esperar, **`both`** se aplica tanto el efecto **`backwards`** como el **`forwards`**:

animation-play-state

Define si la animación se está **ejecutando** o en **pausa**. Una animación en pausa muestra el estado actual de la animación. Cuando se reanuda una animación en pausa, se reinicia desde la posición actual.

La propiedad *animation* abreviada

Afortunadamente, hay una forma abreviada de todas estas propiedades de animación.

```
/* versión larga – longhand */
.verbose {
    animation-name: appear;
    animation-duration: 300ms;
    animation-timing-function: ease-in;
    animation-iteration-count: 1;
    animation-direction: alternate;
    animation-delay: 5s;
    animation-fill-mode: backwards;
}

/* abreviado - shorthand */
.concise {
    animation: appear 300ms ease-in 1 alternate 5s backwards;
}
```

Para declarar múltiples animaciones sobre un elemento, se incluye una agrupación para cada nombre de animación, con cada agrupación abreviada separada por una coma.

```
.target {
    animation:
        animacionUno 300ms ease-in 0s backwards,
        animacionDos 600ms ease-out 1s forwards;
}
```