



Arkaitz Garro

CSS3

Esta página se ha dejado vacía a propósito

CSS 3

Publication date: 14/06/2013

This book was published with *easybook v5.0-DEV*, a free and open-source book publishing application developed by [Javier Eguiluz \(http://javiereguiluz.com\)](http://javiereguiluz.com) using several [Symfony components \(http://components.symfony.com\)](http://components.symfony.com).

Esta página se ha dejado vacía a propósito

Esta obra se publica bajo la licencia *Creative Commons Reconocimiento - No Comercial - Compartir Igual 3.0*, cuyos detalles puedes consultar en <http://creativecommons.org/licenses/by-nc-sa/3.0/es/>.

Puedes copiar, distribuir y comunicar públicamente la obra, incluso transformándola, siempre que cumplas todas las condiciones siguientes:

- Reconocimiento: debes reconocer siempre la autoría de la obra original, indicando tanto el nombre del autor (Arkaitz Garro) como el nombre del sitio donde se publicó originalmente www.arkaitzgarro.com. Este reconocimiento no debe hacerse de una manera que sugiera que el autor o el sitio apoyan el uso que haces de su obra.
- No comercial: no puedes utilizar esta obra con fines comerciales de ningún tipo. Entre otros, no puedes vender esta obra bajo ningún concepto y tampoco puedes publicar estos contenidos en sitios web que incluyan publicidad de cualquier tipo.
- Compartir igual: si alteras o transformas esta obra o si realizas una obra derivada, debes compartir tu trabajo obligatoriamente bajo esta misma licencia.

Esta página se ha dejado vacía a propósito

Capítulo 1 Introducción	11
1.1 Especificación oficial	12
1.2 Nuevas funcionalidades	12
Capítulo 2 Módulo <i>Selectors</i>	15
2.1 Prefijos específicos de navegadores	18
Capítulo 3 Módulo <i>Basic User Interface</i>	21
3.1 <i>box-sizing</i>	21
3.2 <i>overflow-x</i> y <i>overflow-y</i>	23
Capítulo 4 Módulo <i>Color</i>	25
4.1 Propiedad <i>opacity</i>	25
4.2 Función <i>rgba()</i>	26
4.3 Función <i>hsl()</i>	26
4.4 Función <i>hsla()</i>	27
Capítulo 5 Módulo <i>Media Queries</i>	29
5.1 Sintaxis	29
5.2 Operadores lógicos o <i>logical operators</i>	30
Capítulo 6 Módulo <i>Backgrounds and Borders</i>	33
6.1 Bordes	33
6.2 Sombras	36
6.3 Fondos	38
Capítulo 7 Módulo <i>Image Values and Replaced Content</i>	43
7.1 CSS gradients	43
7.2 Degradados lineales	43
7.3 Degradados radiales	47
7.4 Repetición de degradados	49
Capítulo 8 Módulo <i>Multi-column Layout</i>	53
8.1 Propiedad <i>shorthand</i>	54
8.2 Equilibrio de altura	54
8.3 Espacios entre columnas	54
8.4 Trazos entre columnas	55

8.5 Texto sobre columnas	55
Capítulo 9 Módulo <i>Animations</i>.	57
9.1 Configuración	58
9.2 <i>keyframes</i>	58
9.3 Ejemplos	59
Capítulo 10 Módulo <i>Transforms</i>.	65
10.1 Propiedades principales	65
10.2 Funciones de transformación 2D	66
10.3 Ejemplos	67
10.4 Propiedades 3D	67
Capítulo 11 Módulo <i>Transitions</i>	71
11.1 Propiedades "animables"	72
11.2 Propiedad <i>transition-property</i>	72
11.3 Propiedad <i>transition-duration</i>	73
11.4 Propiedad <i>transition-timing-function</i>	73
11.5 Propiedad <i>transition-delay</i>	74
11.6 Propiedad <i>transition</i>	75
11.7 Listas de valores	75
11.8 Finalización de una transición.	76
11.9 Transiciones y JavaScript.	76
Capítulo 12 Módulo <i>Text</i>.	79
12.1 Sombra en textos	79
12.2 Fuentes web	80
Capítulo 13 Ejercicios de CSS 3.	83
13.1 Capítulo 2	83
13.2 Capítulo 3	84
13.3 Capítulo 4	86
13.4 Capítulo 5	87
13.5 Capítulo 6	87
13.6 Capítulo 7	89
13.7 Capítulo 8	91
13.8 Capítulo 10	92

13.9 Ejercicios finales	93
-----------------------------------	----

Esta página se ha dejado vacía a propósito

Capítulo 1

Con la introducción de CSS 3, nunca ha sido tan fácil crear aplicaciones y sitios tan atractivos y completos en HTML. Hay una gran cantidad de extensiones y tecnologías nuevas para CSS 3, entre las que se incluyen las transformaciones en 2D, las transiciones, las transformaciones en 3D y las fuentes web.

El objetivo inicial de CSS, separar el contenido de la forma, se cumplió ya con las primeras especificaciones del lenguaje. Sin embargo, el objetivo de ofrecer un control total a los diseñadores sobre los elementos de la página ha sido más difícil de cubrir. Las especificaciones anteriores del lenguaje tenían muchas utilidades para aplicar estilos a los sitios web, pero los desarrolladores aun continúan usando trucos diversos para conseguir efectos tan comunes o tan deseados como los bordes redondeados o el sombreado de elementos en la página.

CSS 2 ya ofrece un control suficiente para el diseño de los sitios web, pero CSS 3 todavía avanza un poco más en la dirección, de aportar más control sobre los elementos de la página. Así pues, la novedad más importante que aporta CSS 3, de cara a los diseñadores y desarrolladores, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos o hacks, que a menudo complicaban el código, tanto HTML como CSS.

A diferencia de CSS 2, que fue una gran especificación que definía varias funcionalidades, CSS 3 está dividida en varios documentos separados, llamados "módulos". Cada módulo añade nuevas funcionalidades a las definidas en CSS 2, de manera que se preservan las anteriores para mantener la compatibilidad.

Debido a la modularización del CSS 3, diferentes módulos pueden encontrarse en diferentes estados de su desarrollo, existiendo actualmente alrededor de cincuenta módulos publicados. El estado actual del desarrollo de CSS 3 puede ser consultado en <http://www.w3.org/Style/CSS/current-work>.

Mientras CSS 3 se convierte en un estándar, los distintos navegadores pueden incluir o no las mejoras propuestas, o incluso añadir las suyas propias. La web *Can I use...* (<http://caniuse.com/>) incluye un listado completo de funcionalidades soportadas por los distintos navegadores.

Estas son algunas de las nuevas funcionalidades añadidas en CSS 3:

- **Módulo *Selectors Level 3***

- Nuevos selectores por atributos
- Nuevas pseudo-clases
- Nuevo selector adyacente

- **Módulo *Basic User Interface***

- Propiedad box-sizing
- Propiedades nav-top, nav-right, nav-bottom, nav-left
- Propiedades overflow-x, overflow-y

- **Módulo *Color***

- Colores HSL
- Colores HSLA
- Colores RGBA
- Opacidad

- **Módulo *Media Queries***
- **Módulo *Backgrounds and Borders***
 - Propiedad background-origin
 - Propiedad background-clip
 - Propiedad background-size
 - Propiedad background
 - Propiedad border-color
 - Propiedad border-image
 - Propiedad border-radius
 - Propiedad box-shadow
- **Módulo *Image Values and Replaced Content***
 - Degradados lineales
 - Degradados radiales
 - Degradados lineales de repetición
 - Degradados radiales de repetición
- **Módulo *Multi-column Layout***
- **Módulo *Animations***
- **Módulo *Transforms***
- **Módulo *Transitions***
- **Módulo *Fonts Level 3***
- **Módulo *Text Level 3***
 - Propiedad text-shadow
 - Propiedad text-overflow
 - Propiedad word-wrap

Esta página se ha dejado vacía a propósito

Capítulo 2

La versión actual de CSS 3 incluye todos los selectores de CSS 2.1 y añade otras decenas de selectores, pseudo-clases y pseudo-elementos. La lista provisional de novedades y su explicación detallada se puede encontrar en el módulo de selectores de CSS 3 (<http://www.w3.org/TR/css3-selectors/>) .

En primer lugar, CSS 3 añade tres nuevos selectores de atributos:

- `elemento[atributo^="valor"]`, selecciona todos los elementos que disponen de ese atributo y cuyo valor comienza exactamente por la cadena de texto indicada.
- `elemento[atributo$="valor"]`, selecciona todos los elementos que disponen de ese atributo y cuyo valor termina exactamente por la cadena de texto indicada.
- `elemento[atributo*="valor"]`, selecciona todos los elementos que disponen de ese atributo y cuyo valor contiene la cadena de texto indicada.

De esta forma, se pueden crear reglas CSS tan avanzadas como las siguientes:

```
/* Selecciona todos los enlaces que apuntan a una dirección de correo electrónico */  
a[href^="mailto:"] { ... }  
  
/* Selecciona todos los enlaces que apuntan a una página HTML */
```

```
a[href$=".html"] { ... }

/* Selecciona todos los títulos h1 cuyo atributo title contenga la
palabra
    "capítulo" */
h1[title*="capítulo"] { ... }
```

Otro de los nuevos selectores de CSS 3 es el "*selector general de elementos hermanos*", que generaliza el selector adyacente de CSS 2.1. Su sintaxis es `elemento1 ~ elemento2` y selecciona el `elemento2` que es hermano de `elemento1` y se encuentra detrás en el código HTML. En el selector adyacente la condición adicional era que los dos elementos debían estar uno detrás de otro en el código HTML, mientras que ahora la única condición es que uno esté detrás de otro.

Si se considera el siguiente ejemplo:

```
h1 + h2 { ... } /* selector adyacente */
h1 ~ h2 { ... } /* selector general de hermanos */

<h1>...</h1>
<h2>...</h2>
<p>...</p>
<div>
  <h2>...</h2>
</div>
<h2>...</h2>
```

El primer selector (`h1 + h2`) sólo selecciona el primer elemento `<h2>` de la página, ya que es el único que cumple que es hermano de `<h1>` y se encuentra justo detrás en el código HTML. Por su parte, el segundo selector (`h1 ~ h2`) selecciona todos los elementos `<h2>` de la página salvo el segundo. Aunque el segundo `<h2>` se encuentra detrás de `<h1>` en el código HTML, no son elementos hermanos porque no tienen el mismo elemento padre.

Los pseudo-elementos de CSS 2.1 se mantienen en CSS 3, pero cambia su sintaxis y ahora se utilizan `::` en vez de `:` delante del nombre de cada pseudo-elemento:

- `::first-line`, selecciona la primera línea del texto de un elemento.
- `::first-letter`, selecciona la primera letra del texto de un elemento.

- `::before`, selecciona la parte anterior al contenido de un elemento para insertar nuevo contenido generado.
- `::after`, selecciona la parte posterior al contenido de un elemento para insertar nuevo contenido generado.

CSS 3 añade además un nuevo pseudo-elemento:

- `::selection`, selecciona el texto que ha seleccionado un usuario con su ratón o teclado.

Las mayores novedades de CSS 3 se producen en las pseudo-clases, ya que se añaden 12 nuevas, entre las cuales se encuentran:

- `elemento:nth-child(numero)`, selecciona el elemento indicado pero con la condición de que sea el hijo enésimo de su padre. Este selector es útil para seleccionar el segundo párrafo de un elemento, el quinto elemento de una lista, etc.
- `elemento:nth-last-child(numero)`, idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.
- `elemento:empty`, selecciona el elemento indicado pero con la condición de que no tenga ningún hijo. La condición implica que tampoco puede tener ningún contenido de texto.
- `elemento:first-child` y `elemento:last-child`, seleccionan los elementos indicados pero con la condición de que sean respectivamente los primeros o últimos hijos de su elemento padre.
- `elemento:nth-of-type(numero)`, selecciona el elemento indicado pero con la condición de que sea el enésimo elemento hermano de ese tipo.
- `elemento:nth-last-of-type(numero)`, idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.

Algunas pseudo-clases como `:nth-child(numero)` permiten el uso de expresiones complejas para realizar selecciones avanzadas:

```
/* selecciona todos los elementos impares de una lista */
li:nth-child(2n+1) { ... }
/* selecciona todos los elementos pares de una lista */
li:nth-child(2n) { ... }

/* Las siguientes reglas alternan cuatro estilos diferentes para los
```

```
párrafos */
p:nth-child(4n+1) { ... }
p:nth-child(4n+2) { ... }
p:nth-child(4n+3) { ... }
p:nth-child(4n+4) { ... }
```

Empleando la pseudo-clase `:nth-of-type(numero)` se pueden crear reglas CSS que alternen la posición de las imágenes en función de la posición de la imagen anterior:

```
img:nth-of-type(2n+1) { float: right; }
img:nth-of-type(2n)   { float: left; }
```

Otro de los nuevos selectores que incluirá CSS 3 es `:not()`, que se puede utilizar para seleccionar todos los elementos que no cumplen con la condición de un selector:

```
/* selecciona todos los elementos de la página que no sean párrafos */
:not(p) { ... }
/* selecciona cualquier elemento cuyo atributo id no sea "especial" */
:not(#especial) { ... }
```

Aunque todavía faltan muchos años hasta que la versión CSS 3 sustituya a la actual versión CSS 2.1, la mayoría de los navegadores incluyen soporte para varios o casi todos los selectores de CSS 3.

Ejercicio 1

[Ver enunciado \(#ej01\)](#)

Los prefijos específicos de navegador son una manera que tienen los distintos navegadores de añadir o probar nuevas funcionalidades de CSS. Son utilizados para añadir funcionalidades específicas que no forman parte de la especificación oficial o para añadir futuras funcionalidades que no han sido implementadas del todo. Su formato es el siguiente:

```
selector {
    -prefix-property: value;
}
```

Los prefijos para los distintos navegadores son los siguientes:

- `-webkit-`: Android, Chrome, iOS, Safari

- -moz-: Firefox
- -ms-: Internet Explorer
- -o-: Opera

Por ejemplo, para añadir una transición CSS 3 a un elemento, debemos utilizar la propiedad `transition` con los siguientes prefijos:

```
-webkit-transition: all 4s ease;  
-moz-transition: all 4s ease;  
-ms-transition: all 4s ease;  
-o-transition: all 4s ease;  
transition: all 4s ease;
```

Estos prefijos pretenden ser temporales hasta que el navegador implemente completamente la funcionalidad o forme parte del estándar oficialmente. Los navegadores que utilicen prefijos para ciertas propiedades, las tendrán en cuenta e ignorarán el resto, y los que no hagan uso de ellas tendrán en cuenta la última regla indicada.

Esta página se ha dejado vacía a propósito

Capítulo 3

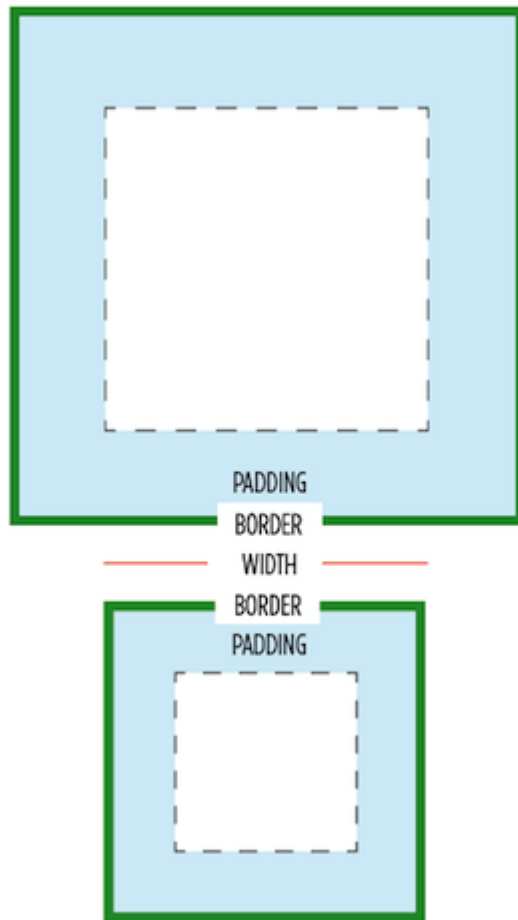
La especificación oficial y el estado actual de desarrollo del módulo *Basic User Interface* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-ui/>.

La propiedad CSS `box-sizing` se usa para alterar el *box model* o modelo de caja por defecto usado para calcular alturas y anchuras de elementos. Es posible usar esta propiedad para emular el comportamiento de navegadores que no soportan correctamente la especificación de *box model* de CSS.

El *box model* o modelo de caja en CSS funciona de la siguiente manera:

$\text{width} + \text{padding} + \text{border} = \text{actual visible/rendered width of box}$

$\text{height} + \text{padding} + \text{border} = \text{actual visible/rendered height of box}$



La sintaxis de la propiedad `box-sizing` es la siguiente:

	<code>box-sizing</code>
Valores	<code>content-box</code> <code>padding-box</code> <code>border-box</code>
Se aplica a	Todos los elementos que aceptan <code>width</code> o <code>height</code>
Valor inicial	<code>content-box</code>
Descripción	Indica el componente del modelo de caja que se toma como referencia para calcular el alto o ancho de un elemento

Los valores que puede tomar esta propiedad son los siguientes:

- `content-box`: este es el estilo por defecto especificado en el estándar de CSS. Las propiedades `width` y `height` se miden incluyendo sólo el contenido, pero no el borde, margen o relleno.

- padding-box: las propiedades width y height incluyen el tamaño del relleno pero no incluyen el borde ni margen.
- border-box: las propiedades width y height incluyen el relleno y el borde, pero no el margen.

```
/* support Firefox, WebKit, Opera and IE8+ */
.example {
    box-sizing: border-box;
    width: 500px;
}
```

Las propiedades `overflow` son utilizadas para controlar el comportamiento del contenido dentro de un elemento de bloque. Especifican si recortan el contenido, muestran una barra de desplazamiento o contenido "se sale" de un elemento de bloque cuando se desborda en los bordes izquierdo y derecho (x) o superior e inferior (y).

	overflow-x overflow-y
Valores	visible hidden scroll auto inherit
Se aplica a	Todos los elementos de bloque e <code>inline-block</code>
Valor inicial	auto
Descripción	Indica el comportamiento del contenido al superar el tamaño de su contenedor

Esta página se ha dejado vacía a propósito

Capítulo 4

La especificación oficial y el estado actual de desarrollo del módulo *Color* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-color/>.

La propiedad de CSS `color` denota un color en el espacio de color sRGB. Un color puede ser descrito utilizando una *keyword*, el sistema de coordenadas cúbicas RGB o el sistema de coordenadas cilíndricas HSL. Hay que tener en cuenta que la lista de valores de color aceptados se ha ampliado en la evolución de la especificación, que culmina con los colores CSS 3.

Aunque los valores de color CSS están definidos con precisión, pueden aparecer de manera distinta en diferentes dispositivos. La mayoría de ellos no están calibrados, y algunos navegadores no soportan los distintos perfiles de color.

Una de las características más esperadas de CSS 3 ha sido `opacity`. Gracias a esta propiedad, podemos definir el grado de transparencia de un elemento.

	<code>opacity</code>
Valores	<code><alphavalue></code> <code>inherit</code>
Se aplica a	Todos los elementos.
Valor inicial	1

	opacity
Descripción	Especifica la transparencia de un elemento.

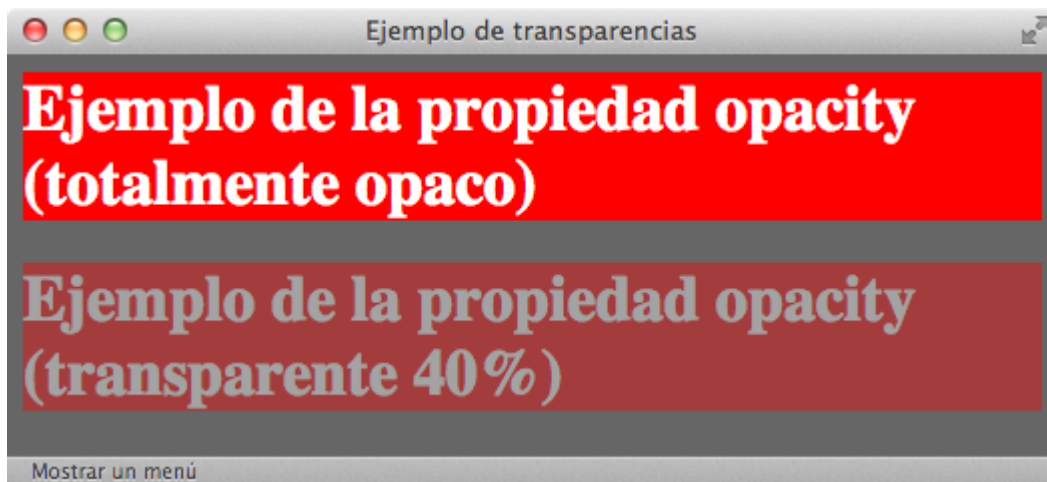


Figura 4.1 Ejemplo de transparencia aplicada a un elemento

Los colores pueden ser definidos gracias al modelo *Red-Green-Blue-alpha (RGBA)* utilizando la notación funcional `rgba()`. **RGBA** extiende el modelo de color RGB para incluir el canal alfa, lo que permite la especificación de la transparencia de un color. `a` significa `opacity`: `0.0=transparent`; `1.0=opaque`;

```

rgba(255,0,0,0.1)    /* 10% opaque red */
rgba(255,0,0,0.4)    /* 40% opaque red */
rgba(255,0,0,0.7)    /* 70% opaque red */
rgba(255,0,0, 1)     /* full opaque red */

```

Los colores también se puede definir gracias el modelo *Hue-Saturation-Lightness (HSL)* utilizando la notación funcional `hsl()`. La ventaja de **HSL** sobre RGB es que es más intuitivo; se pueden "adivinar" los colores que se quieren, y después ajustarlos. También es más sencillo para crear conjuntos de colores combinados (manteniendo el mismo tono y variando la luminosidad/oscuridad y saturación).

Hue (tono) es representado como un ángulo del círculo de color. Este ángulo viene dado como un `<number>` sin unidades. Por definición `red=0=360` y los otros colores se extienden alrededor del círculo, por lo tanto `green=120`, `blue=240`, etc.

La saturación y la luminosidad son representados como porcentajes. 100% es saturación completa y 0% es escala de grises. 100% en la luminosidad es blanco, 0% es negro y 50% es "normal".

```
hsl(0, 100%,50%)    /* red */
hsl(30, 100%,50%)
hsl(60, 100%,50%)
hsl(90, 100%,50%)
hsl(120,100%,50%)   /* green */
hsl(150,100%,50%)
hsl(180,100%,50%)
hsl(210,100%,50%)
hsl(240,100%,50%)   /* blue */
hsl(270,100%,50%)
hsl(300,100%,50%)
hsl(330,100%,50%)
hsl(360,100%,50%)   /* red */

hsl(120,100%,25%)   /* dark green */
hsl(120,100%,50%)   /* green */
hsl(120,100%,75%)   /* light green */

hsl(120,100%,50%)   /* green */
hsl(120, 67%,50%)
hsl(120, 33%,50%)
hsl(120, 0%,50%)

hsl(120, 60%,70%)   /* pastel green */
```

Los colores pueden definirse también utilizando la función `hsla()`. **HSLa** extiende el modelo de color HSL visto anteriormente para incluir el canal alfa, lo que permite definir la opacidad del color. `a` significa `opacity`: 0.0=transparent; 1.0=opaque;.

```
hsla(240,100%,50%,0.05) /* 5% opaque blue */
hsla(240,100%,50%, 0.4) /* 40% opaque blue */
hsla(240,100%,50%, 0.7) /* 70% opaque blue */
hsla(240,100%,50%, 1)   /* full opaque blue */
```

Esta página se ha dejado vacía a propósito

Capítulo 5

La especificación oficial y el estado actual de desarrollo del módulo *Media Queries* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-mediaqueries/>.

Un ***Media Query*** consiste en realizar una consulta sobre un tipo de medio y al menos una expresión que limita el alcance de las hojas de estilo en función a características del dispositivo, como pueden ser el ancho, alto o color. Añadidos a CSS 3, los *Media Queries* permiten que la presentación del contenido se adapte a un rango específico de dispositivos sin tener que cambiar el contenido en sí.

Los *Media Queries* pueden contener una o más expresiones, expresadas como funciones multimedia, que devuelven *true* o *false*. El resultado del *query* o consulta devuelve *true* si el *media type* especificado en el *Media Query* coincide con el tipo de dispositivo en que el documento está siendo mostrado y todas las expresiones en el *Media Query* devuelven *true*.

Cuando un *Media Query* devuelve *true*, la correspondiente hoja de estilo es aplicada, siguiendo las reglas habituales de CSS. Las hojas de estilo con *Media Queries* adjuntos a los *tags* <link> se descargan siempre, incluso si sus *Media Queries* resultan *false* (sin embargo, en este caso, no se aplicarán).

Se pueden crear *Media Queries* complejos utilizando operadores lógicos, incluyendo *not*, *and* y *only*. El operador *and* es usado para combinar múltiples características en un sólo *Media Query*, requiriendo que cada función devuelva *true* para que el *Query* también lo sea. El operador *not* se utiliza para negar un *Media Query* completo y el operador *only* se usa para aplicar un estilo sólo si el *Query* completo es correcto.

Además, se pueden combinar múltiples *Media Queries* separados por comas en una lista; si alguno de los *Queries* devuelve *true*, toda la consulta devolverá **true*. Esto es equivalente a un operador *or*.

El keyword *and* se usa para combinar múltiples características, así como combinar éstos con tipos de medios. Un *Media Query* básico sería:

```
| @media (min-width: 700px) { ... }
```

Sin embargo, si quisiéramos que esto sólo se aplicara si la pantalla está en modo apaisado, se usaría el operador *and*:

```
| @media (min-width: 700px) and (orientation: landscape) { ... }
```

Si además, sólo quisiéramos que esto se aplicara si el dispositivo fuera un *media type TV*:

```
| @media tv and (min-width: 700px) and (orientation: landscape) { ... }
```

Cuando se utilizan las listas separadas por comas en los *Media Queries*, si algunas de las *Media Queries* devuelven *true*, los estilos se aplican al documento. Cada *Media Query* separado por comas en la lista se trata como un *Query* individual, y cualquier operador aplicado a un *Media Query* no afecta a los demás. Esto significa que los *Media Queries* separados por comas puede dirigirse a diferentes *media features*, *types* o *states*.

Por ejemplo, si quisiéramos aplicar un conjunto de estilos si el dispositivo de visualización tiene un mínimo de 700px o está en modo apaisado:

```
| @media (min-width: 700px), handheld and (orientation: landscape) { ... }
```

La *keyword* `not` se aplica al *Media Query* en su totalidad y devuelve *true* si el *Media Query* devuelve *false* (como `monochrome` en una pantalla a color). Este *keyword* no se puede utilizar para negar una característica en concreto, sino la consulta completa. Por ejemplo:

```
| @media not all and (monochrome) { ... }
```

Esto significa que el *Query* es evaluado de esta manera:

```
| @media not (all and (monochrome)) { ... }
```

en vez de así:

```
| @media (not all) and (monochrome) { ... }
```

Por ejemplo, este otro *Media Query*:

```
| @media not screen and (color), print and (color)
```

Se evalúa así:

```
| @media (not (screen and (color))), print and (color)
```

El *keyword* `only` previene a los navegadores que no soportan *Media Queries*:

```
| <link rel="stylesheet" media="only screen and (color)"  
| href="example.css" />
```

Ejercicio 2

[Ver enunciado \(#ej02\)](#)

Esta página se ha dejado vacía a propósito

Capítulo 6

La especificación oficial y el estado actual de desarrollo del módulo *Backgrounds and Borders* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-background/>.

La propiedad `border-radius` permite definir *esquinas redondeadas* en los elementos de una manera muy sencilla, sin necesidad de utilizar imágenes o una estructura de `div` muy compleja. Las *esquinas redondeadas* pueden ser creadas de manera independiente utilizando las cuatro propiedades individuales `border-*-radius` (`border-bottom-left-radius`, `border-bottom-right-radius`, `border-top-left-radius` y `border-top-right-radius`). La curva o radio de cada esquina se define usando uno o dos radios que definen su forma. Si únicamente definimos un radio, tenemos un **círculo**, si definimos dos radios tenemos una **elipse**.

```
border-*-radius: [ <length> | <%> ] [ <length> | <%> ]?
```

```
border-top-left-radius: 10px 5px;  
border-bottom-right-radius: 10% 5%;  
border-top-right-radius: 10px;
```

Las siguientes imágenes muestran un ejemplo de cómo pueden definirse los diferentes radios de un borde:

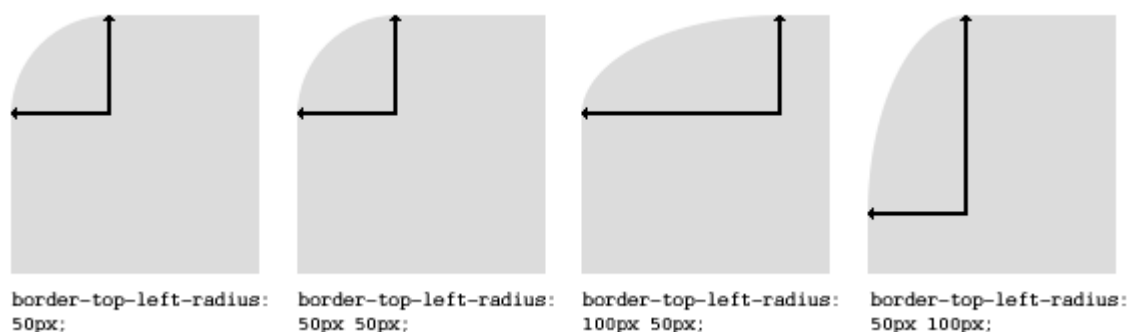


Figura 6.1 Diferentes radios de borde








Esta propiedad es una forma abreviada (*shorthand*) de establecer las cuatro propiedades `border-top-left-radius`, `border-top-right-radius`, `border-bottom-right-radius` y `border-bottom-left-radius`.

	<code>border-radius</code>
Valores	[<length> <percentage>]{1,4} [/ [<length> <percentage>]{1,4}]?
Se aplica a	Todos los elementos, pero los navegadores no están obligados a aplicarlo a los elementos <code>table</code> y <code>inline-table</code> cuando <code>border-collapse</code> es <code>collapse</code> . El comportamiento de los elementos internos de una tabla no está definido por el momento.
Valor inicial	<code>border-bottom-left-radius: 0</code> , <code>border-bottom-right-radius: 0</code> , <code>border-top-left-radius: 0</code> , <code>border-top-right-radius: 0</code>
Descripción	Define el radio de las esquinas

Si se define el primer conjunto de valores, éste indica el radio horizontal de las esquinas. El segundo conjunto de valores, hace referencia al radio vertical. Si únicamente se define el primer conjunto de valores, hacer referencia al radio horizontal y vertical. El valor del radio puede ser:

- `<length>`: se puede expresar en cualquier unidad permitida por los tipos de datos del `<length>` en CSS. Los valores negativos no son válidos.

- `<percentage>`: los porcentajes para el eje horizontal se refieren a la anchura de la caja y los porcentajes para el eje vertical se refieren a la altura. Los valores negativos no son válidos.

radius		Es un <code><length></code> o <code><percentage></code> que indica un radio a utilizar para el borde en cada esquina del elemento.
top-left-and-bottom-right		Es un <code><length></code> o <code><percentage></code> que indica un radio a utilizar para el borde en las esquinas superior-izquierda e inferior-derecha del elemento.
top-right-and-bottom-left		Es un <code><length></code> o <code><percentage></code> que indica un radio a utilizar para el borde en las esquinas superior-derecha e inferior-izquierda del elemento.
top-left		Es un <code><length></code> o <code><percentage></code> que indica un radio a utilizar para el borde en la esquina superior-izquierda del elemento.
top-right		Es un <code><length></code> o <code><percentage></code> que indica un radio a utilizar para el borde en la esquina superior-derecha del elemento.
bottom-right		Es un <code><length></code> o <code><percentage></code> que indica un radio a utilizar para el borde en la esquina inferior-derecha del elemento.
bottom-left		Es un <code><length></code> o <code><percentage></code> que indica un radio a utilizar para el borde en la esquina inferior-izquierda del elemento.
inherit		Es una palabra clave que indica que los cuatro valores se heredan del valor calculado del elemento padre.

```
border-radius: 5px 10px 5px 10px / 10px 5px 10px 5px;  
border-radius: 5px;  
border-radius: 5px 10px / 10px;
```

Ejercicio 6

[Ver enunciado \(#ej06\)](#)

La propiedad `box-shadow` es otra de las características más esperadas de CSS 3. Permite aplicar de una manera muy sencilla sombras a los elementos (internas o externas), especificando el color, la separación y la definición.

La propiedad `box-shadow` acepta lista de valores de sombras separados por comas, cada una de ellas con una longitud de 2 a 4 valores (es posible especificar la separación horizontal, vertical, la definición de la sombra y la distancia de la misma con respecto al borde), un color opcional y de finir si la sombra es interna o externa.

	box-shadow
Valores	none <shadow> [, <shadow>]*
Se aplica a	Todos los elementos.
Valor inicial	none
Descripción	Define el radio de las esquinas

Donde <shadow>:

```
<shadow> = inset? && [ <length>{2,4} && <color>? ]
```

Los posibles valores de <length> son:

- El primer valor indica la separación horizontal de la sombra. Un valor positivo desplaza la sombra hacia la derecha de la caja, mientras que un valor negativo la desplazará hacia la izquierda.
- El segundo valor indica la separación vertical de la sombra. Un valor positivo desplaza la sombra debajo de la caja, mientras que un valor negativo la desplazará hacia arriba.

- El tercer valor es el radio del desenfoque. Si el valor es 0 (no se aceptan valores negativos) la sombra será muy nítida. Para valores mayores que 0, la sombra se va difuminando.
- El cuarto valor es la distancia de propagación. Un valor positivo hace que la sombra se expanda en todas las direcciones según el valor. Valores negativos causan que la sombra se contraiga.

```
box-shadow: 10px 10px;
box-shadow: 10px 10px 5px #888;
box-shadow: inset 2px 2px 2px 2px black;
box-shadow: 10px 10px #888, -10px -10px #f4f4f4, 0px 0px 5px 5px #cc6600;
```

El siguiente diagrama muestra un ejemplo del efecto de aplicar los efectos de desenfoque y propagación de la sombra:

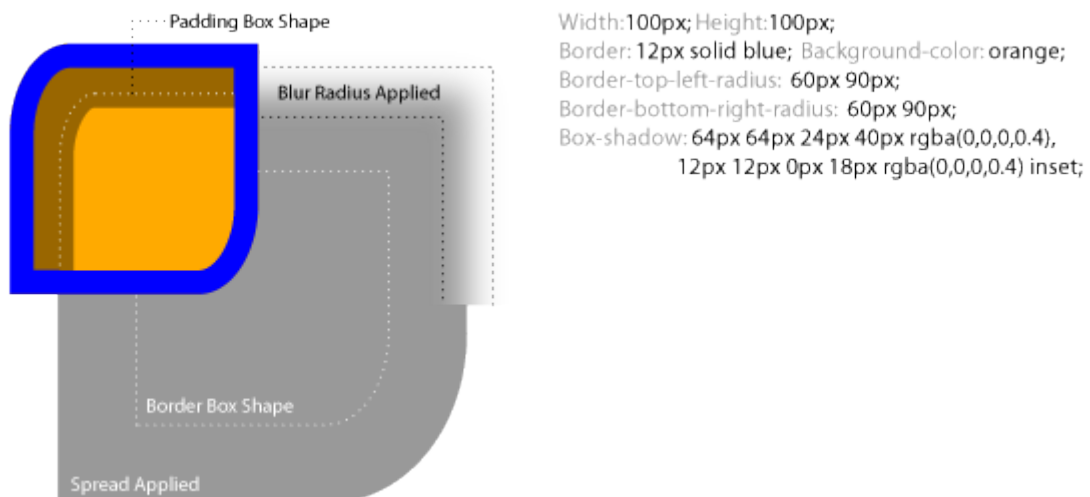


Figura 6.2 Ejemplo de aplicar desenfoque y propagación

Los efectos de sombra son aplicados de arriba hacia abajo, es decir, la primera sombra de la lista está sobre las demás. Las sombras no influyen en la estructura del documento y pueden solaparse entre ellas. En terminos de pila y orden de pintado, las sombras externas son pintadas debajo del fondo del elemento, y las sombras internas son pintadas encima del fondo del elemento.

Ejercicio 7

[Ver enunciado \(#ej07\)](#)

CSS 3 añade nuevas posibilidades a la hora de trabajar con imágenes de fondo. Ahora podemos definir varias imágenes de fondo y especificar su tamaño.

Con CSS3 se pueden aplicar **fondos múltiples** a los elementos. Estos se colocan en **capas** una encima de la otra con el primer fondo en la parte superior y el último en la parte posterior. Sólomente el último fondo puede incluir un color de fondo.

Un ejemplo muy sencillo:

Esta caja tiene dos imágenes de fondo, la primera es una oveja y la segunda es la hierba con el cielo de fondo.

```
#background {  
    width: 500px;  
    height: 250px;  
    background-image: url(sheep.png), url(betweengrassandsky.png);  
    background-position: center bottom, left top;  
    background-repeat: no-repeat;  
}
```

Las múltiples imágenes de fondo pueden especificarse indicando todas las imágenes de fondo separadas por comas, en la propiedad `background-image`, donde cada una de las imágenes crea una nueva capa.

	background-image
Valores	<bg-image> [, <bg-image>]*
Se aplica a	Todos los elementos.
Valor inicial	Ninguno
Descripción	Especifica los fondos de un elemento.

Podemos definir la posición de cada uno de los fondo a través de la propiedad `background-position`, utilizando las mismas palabras clave definidas en CSS 2.1.

	background-position
Valores	<position> [, <position>]*
Se aplica a	Todos los elementos.
Valor inicial	0% 0%
Descripción	Especifica la posición de las imágenes de fondo definidas.

Cada uno de los valores separados por comas hacer referencia a las imágenes de fondo definidas en la propiedad `background-image`. Si se definen más valores que imágenes de fondo, los valores sobrantes son ignorados. Por ejemplo, si definimos tres imágenes de fondo, y cinco posiciones, las dos últimas serán ignoradas.

De igual manera, si se especifican menos posiciones que imágenes de fondo, la lista de valores se repite desde el primer elemento, tantas veces como sea necesario.

Otra de las propiedades introducidas en CSS 3 ha sido `background-size`. Esta propiedad permite ajustar el **tamaño de las imágenes de fondo**, en lugar del comportamiento predeterminado de mosaico, utilizando medidas estándar, porcentajes o las palabras reservadas `contain` y `cover`.

Un ejemplo muy sencillo:

```
background-size: auto;
background-size: 275px 125px;
#background1 {
    background-size: auto;
}

#background2 {
    background-size: 275px 125px;
}
```

Como en otras muchas propiedades de `background`, es posible especificar los distintos valores separados por comas.

	background-size
Valores	<bg-size> [, <bg-size>]*
Se aplica a	Todos los elementos.
Valor inicial	Ver propiedades individuales
Descripción	Especifica los tamaños de fondos de un elemento.

Donde:

<bg-size> = [<length> | <percentage> | **auto**]{1,2} | cover | contain

Ejemplo:

```
background-size: 200px;  
background-size: 200px 100px;  
background-size: 200px 100px, 400px 200px;  
background-size: auto 200px;  
background-size: 50% 25%;  
background-size: contain;  
background-size: cover;
```

Al especificar los valores de <length> o <percentage>, su comportamiento es el esperado y es el que conocemos por ejemplo con las imágenes. Si especificamos un valor, este hace referencia al ancho (manteniendo el alto proporcional) y si especificamos dos valores, estos hacen referencia al ancho y alto respectivamente.

Además de los valores <length> y <percentage> que se comportan de la manera que conocemos hasta ahora, la propiedad background-size admite dos valores especiales: contain y cover.

- **contain**: en este caso, la imagen es redimensionada manteniendo sus proporciones originales, de tal manera que cada lado sea tan grande como sea posible mientras que no exceda la longitud del lado correspondiente del contenedor. La imagen nunca excede el tamaño de su contenedor, por lo que si no coincide con el tamaño de su contenedor, pueden existir zonas del fondo que no queden cubiertas por la imagen.
- **cover**: en este otro caso, la imagen es redimensionada manteniendo sus proporciones originales, de tal manera que el área completa del

fondo es siempre cubierta por la imagen de fondo. En este caso, el tamaño de la imagen siempre es igual o superior al área del fondo a cubrir, pudiendo existir zonas de la imagen que no sean visibles.

Es posible utilizar la propiedad *shorthand* `background` para especificar los distintos fondos y sus propiedades en una misma sentencia, indicando los distintos fondos separados por comas. Si se desea especificar un color de fondo, éste únicamente puede ser definido en el último valor de la propiedad.

	background
Valores	[<bg-layer> ,]* <final-bg-layer>
Se aplica a	Todos los elementos.
Valor inicial	Ver propiedades individuales
Descripción	Especifica los fondos de un elemento.

Donde:

```
<bg-layer> = <bg-image> || <bg-position> [ / <bg-size> ]? ||  
<repeat-style> || <attachment> || <box>{1,2}  
  
<final-bg-layer> = <bg-image> || <bg-position> [ / <bg-size> ]? ||  
<repeat-style> || <attachment> || <box>{1,2} || <background-color>
```

Ejemplo:

```
background: url(sheep.png) center bottom no-repeat,  
url(betweengrassandsky.png) left top no-repeat;
```

Ejercicio 8

[Ver enunciado \(#ej08\)](#)

Esta página se ha dejado vacía a propósito

Capítulo 7

La especificación oficial y el estado actual de desarrollo del módulo *Image Values and Replaced Content* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-images/>.

Los degradados CSS son los nuevos tipo de `<image>` añadidos en el módulo de imagen de CSS3. Utilizar los degradados de CSS permite mostrar **transiciones suaves entre dos o más colores**; esto permite a su vez evistar el uso de imágenes para lograr estos efectos, lo que reduce mucho el tiempo de descarga y el uso de ancho de banda. Además, debido a que el degradado es generado por el navegador, los objetos con degradados se ven mejor cuando se acercan, y se puede ajustar el diseño de una manera mucho más flexible.

Los navegadores soportan dos tipos de degradados: **lineal**, que se define con la función `linear-gradient` y **radial**, definido con `radial-gradient`.

Para crear un degradado lineal, se establece un punto de partida y una dirección (especificado como un ángulo) a lo largo del cual se aplica dicho degradado.

Screen Shot	Live Demo
	

/* The old syntax, deprecated, but still needed, prefixed, for WebKit-based and old browsers */

```
background: -prefix-linear-gradient(top, blue, white);
```

/* The new syntax needed by standard-compliant browsers (Opera 12.1, IE 10, Fx 16 onwards), without prefix */

```
background: linear-gradient(to bottom, blue, white);
```

Cambiando el mismo degradado para que se represente de izquierda a derecha:

Screen Shot	Live Demo
	

/* The old syntax, deprecated, but still needed, prefixed, for WebKit-based and old browsers */

```
background: -prefix-linear-gradient(left, blue, white);
```

/* The new syntax needed by standard-compliant browsers (Opera 12.1, IE 10, Fx 16 onwards), without prefix */

```
background: linear-gradient(to right, blue, white);
```

Y en diagonal:

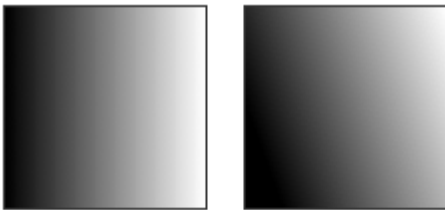
Screen Shot	Live Demo
	

```
/* The old syntax, deprecated, but still needed, prefixed, for
WebKit-based and old browsers */
background: -prefix-linear-gradient(left top, blue, white);

/* The new syntax needed by standard-compliant browsers (Opera 12.1, IE
10, Fx 16 onwards), without prefix */
background: linear-gradient(to bottom right, blue, white);
```

Si no se especifica ningún ángulo, se determina uno de manera automática en función de la dirección dada. Si se desea más control sobre la dirección del degradado, se puede establecer específicamente dicho ángulo.

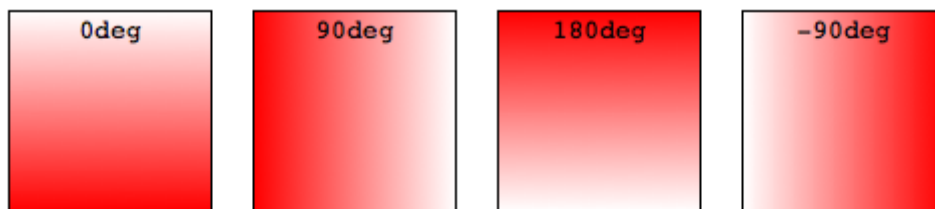
Por ejemplo, aquí tenemos dos degradados, el primero con una dirección hacia la derecha, y el segundo con un ángulo de 70 grados.



El de la derecha utiliza CSS de esta manera:

```
background: linear-gradient(70deg, black, white);
```

El ángulo se especifica como un ángulo entre una línea horizontal y la línea de degradado, yendo hacia la izquierda. En otras palabras, 0deg crea un degradado vertical desde la parte inferior a la parte superior, y 90deg genera un degradado horizontal de izquierda a derecha.

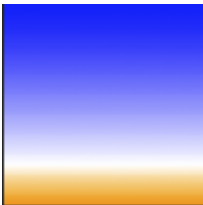



```
background: linear-gradient(<angle>, red, white);
```



Los *color stops* son los puntos de la línea de degradado que tendrán un color específico en dicho lugar. La ubicación se puede especificar con un porcentaje de la longitud de la línea o como una longitud absoluta. Se puede especificar tantas paradas de color como se quiera con el fin de conseguir el efecto deseado.

0% representaría el punto de partida y 100% el punto final. Sin embargo, se pueden utilizar los valores fuera de ese rango si es necesario para obtener el efecto deseado.

Ejemplo de tres paradas de color:

Screen Shot	Live Demo
	
<pre>/* The old syntax, deprecated, but still needed, prefixed, for WebKit-based and old browsers */ background: -prefix-linear-gradient(top, blue, white 80%, orange); /* The new syntax needed by standard-compliant browsers (Opera 12.1, IE 10, Fx 16 onwards), without prefix */ background: linear-gradient(to bottom, blue, white 80%, orange);</pre>	

Ejemplo de variedad de colores:

Screen Shot	Live Demo
	
<pre>/* The old syntax, deprecated, but still needed, prefixed, for WebKit-based and old browsers */ background: -prefix-linear-gradient(left, red, orange, yellow, green, blue);</pre>	

```
/* The new syntax needed by standard-compliant browsers (Opera 12.1, IE
10, Fx 16 onwards), without prefix */
background: linear-gradient(to right, red, orange, yellow, green, blue);
```

Los degradados admiten transparencia. Se puede utilizar, por ejemplo, con múltiples fondos o para crear efectos en imágenes de fondo. Por ejemplo:



```
/* The old syntax, deprecated, but still needed, prefixed, for
WebKit-based and old browsers */
background: linear-gradient(left, rgba(255,255,255,0),
rgba(255,255,255,1)), url(http://foo.com/image.jpg);
```

```
/* The new syntax needed by standard-compliant browsers (Opera 12.1, IE
10, Fx 16 onwards), without prefix */
background: linear-gradient(to right, rgba(255,255,255,0),
rgba(255,255,255,1)), url(http://foo.com/image.jpg);
```

Los **degradados radiales** se especifican usando la función `radial-gradient`. La sintaxis es similar a los degradados lineales, excepto en que se puede especificar la forma final del degradado (circular o elíptico) así como el tamaño. Por defecto, la forma final es una elipse con las mismas proporciones que la caja contenedora.

Un ejemplo sería:

Screen Shot	Live Demo
	

```
background: radial-gradient(red, yellow, rgb(30, 144, 255));
```

Y un segundo ejemplo con localizaciones específicas:

Screen Shot	Live Demo
	

```
background: radial-gradient(red 5%, yellow 25%, #1E90FF 50%);
```

Ésta es una de las áreas en las que los degradados radiales se diferencian de los lineales. Se puede proporcionar un valor de tamaño que especifica el punto que define el tamaño del círculo o elipse.

Ejemplos para elipses

Ejemplo con `closest-side` (el tamaño es establecido por la distancia entre el punto de partida o centro y el lugar más próximo a la caja contenedora):

Screen Shot	Live Demo
	

```
background: radial-gradient(ellipse closest-side, red, yellow 10%,  
#1E90FF 50%, white);
```

Ejemplo similar con `farthest-corner`:

Screen Shot	Live Demo
	


```
background: radial-gradient(ellipse farthest-corner, red, yellow 10%,  
#1E90FF 50%, white);
```

Ejemplos para círculos

Ejemplo con closest-side:

Screen Shot	Live Demo
	

```
background: radial-gradient(circle closest-side, red, yellow 10%,  
#1E90FF 50%, white);
```

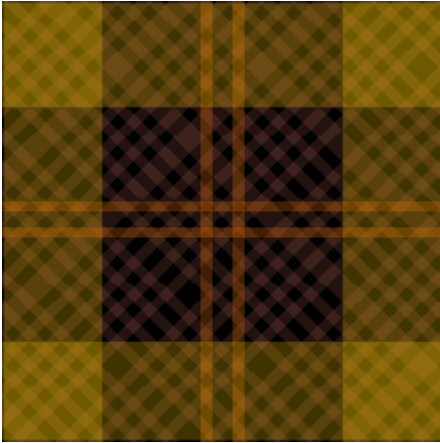
Las propiedades `linear-gradient` y `radial-gradient` no admiten automáticamente la repetición de los *color stops*. Sin embargo, las propiedades `repeating-linear-gradient` y `repeating-radial-gradient` sí ofrecen esta funcionalidad.

Este ejemplo utiliza `repeating-linear-gradient`:

Screen Shot	Live Demo
	

```
background: repeating-linear-gradient(-45deg, red, red 5px, white 5px,  
white 10px);
```

Otro ejemplo con esta misma propiedad:



```
background-color: #000;
background-image: repeating-linear-gradient(90deg, transparent,
transparent 50px,
    rgba(255, 127, 0, 0.25) 50px, rgba(255, 127, 0, 0.25) 56px,
transparent 56px, transparent 63px,
    rgba(255, 127, 0, 0.25) 63px, rgba(255, 127, 0, 0.25) 69px,
transparent 69px, transparent 116px,
    rgba(255, 206, 0, 0.25) 116px, rgba(255, 206, 0, 0.25) 166px),
repeating-linear-gradient(0deg, transparent, transparent 50px, rgba(255,
127, 0, 0.25) 50px,
    rgba(255, 127, 0, 0.25) 56px, transparent 56px, transparent 63px,
rgba(255, 127, 0, 0.25) 63px,
    rgba(255, 127, 0, 0.25) 69px, transparent 69px, transparent 116px,
rgba(255, 206, 0, 0.25) 116px,
    rgba(255, 206, 0, 0.25) 166px),
repeating-linear-gradient(-45deg, transparent, transparent 5px,
rgba(143, 77, 63, 0.25) 5px,
    rgba(143, 77, 63, 0.25) 10px),
repeating-linear-gradient(45deg, transparent, transparent 5px, rgba(143,
77, 63, 0.25) 5px,
    rgba(143, 77, 63, 0.25) 10px);
```

Y un ejemplo de repetición de un degradado radial:

Screen Shot	Live Demo

```
background: repeating-radial-gradient(black, black 5px, white 5px, white  
10px);
```

Esta página se ha dejado vacía a propósito

Capítulo 8

La especificación oficial y el estado actual de desarrollo del módulo *Multi-column Layout* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-multicol/>.

La propiedad `multi-column` permite crear estructuras multicolumna con nuestros textos de dos maneras diferentes: definiendo el número de columnas (con la propiedad `column-count`), o definiendo un ancho para dichas columnas (con la propiedad `column-width`). Es posible especificar el espacio entre columnas con la propiedad `column-gap`.

Un ejemplo de estructura multicolumna utilizando la propiedad `column-count`:

```
| div { column-count:2; }
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Un ejemplo de estructura multicolumna utilizando la propiedad `column-width`:

```
div { column-width:10em; }
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Como los valores de las propiedades anteriores no se superponen, a menudo es conveniente utilizar la propiedad *shorthand* `columns`.

	<code>columns</code>
Valores	<code><column-width> <column-count></code>
Se aplica a	Todos los elementos de bloque (excepto tablas), celdas de tabla y elementos inline-block.
Valor inicial	<code>none</code>
Descripción	Especifica el número de columnas y/o ancho de las mismas.

La especificación de CSS3 requiere que la **alturas de las columnas** sea equilibrada. Esto es, el navegador automáticamente establece el máximo de altura de la columna de manera que las alturas de los contenidos de las columnas sean aproximadamente iguales.

Sin embargo, en algunas situaciones, es útil establecer explícitamente el máximo de altura. Si la altura está limitada mediante las propiedades CSS `height` o `max-height` en un bloque multi-columna, cada columna puede "crecer" hasta la máxima altura y no más allá antes de añadir una nueva columna.

Siempre existe un espacio entre columnas. El espacio recomendado es 1em. Este tamaño puede ser cambiado aplicando la propiedad `column-gap`.

	column-gap
Valores	<length> normal
Se aplica a	Elementos multi-columna
Valor inicial	normal
Descripción	Especifica el espacio entre columnas

En el espacio que existe entre las columnas, es posible dibujar un trazo, como si de un borde se tratara.

	column-rule
Valores	<column-rule-width> <column-rule-style> [<column-rule-color> transparent]
Se aplica a	Elementos multi-columna
Valor inicial	transparent
Descripción	Especifica el trazo a dibujar en el espacio entre columnas.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Es posible mostrar un elemento, inicialmente incluido en el texto formateado en columnas, para que ocupe el ancho completo de las columnas, utilizando la propiedad `column-span`.

	column-span
Valores	none all]
Se aplica a	Elementos de bloque, excepto los flotantes y los definidos con posición absoluta.

	<code>column-span</code>
Valor inicial	none
Descripción	Especifica cuantas columnas ocupa el elemento.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Capítulo 9

La especificación oficial y el estado actual de desarrollo del módulo *Animations* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-animations/>.

Las **animaciones CSS 3** hacen posible animar transiciones de una configuración de estilo de CSS a otra. Las animaciones constan de dos componentes: un estilo que describe la animación CSS y un conjunto de fotogramas que indican el comienzo y final de la animación, así como los posibles puntos de ruta intermedios durante la animación.

Hay tres **ventajas clave** en las animaciones CSS sobre las técnicas de animación tradicionales ejecutadas a través de *scripts*:

- Son fáciles de usar para las animaciones simples; puedes crearlas sin tener la necesidad de conocer el funcionamiento de JavaScript.
- Las animaciones funcionan bien, incluso bajo carga de sistema medio. Las animaciones simples pueden, a menudo, ejecutarse de manera pobre en JavaScript (a menos que estén bien hechas). El motor de renderizado puede no reproducir ciertos fotogramas y utilizar otras técnicas para mantener la reproducción lo más suave posible.
- Dejar al navegador controlar la secuencia de animación permite a éste optimizar el rendimiento y la eficiencia a través de, por ejemplo, la reducción de la frecuencia de actualización de las animaciones en ejecución en pestañas que no están visibles en ese momento.

Para crear una **secuencia de animación CSS**, se requiere dar estilo al elemento que se quiere animar con la **propiedad de animación** o sus **sub-propiedades**. Esto permite configurar los tiempos y duración de la animación, así como otros detalles de cómo la secuencia de animación debe progresar. Esto, sin embargo, no configura la apariencia real de la animación, que se establece utilizando los keyframes explicados en el apartado siguiente.

Las **sub-propiedades** de la propiedad animación son:

- `animation-delay`: configura el retraso entre lo que tarda el elemento en cargarse y el comienzo de la secuencia de animación.
- `animation-direction`: configura si la animación debe alternar dirección en cada reproducción a través de la secuencia o volver al punto de inicio y repetirse.
- `animation-duration`: configura el tiempo que tarda la animación en completar un ciclo.
- `animation-iteration-count`: configura el número de veces que la animación debe repetirse, puede establecerse el valor `infinite` para repetir la animación de forma indefinida.
- `animation-name`: especifica el nombre de los keyframes que describen los fotogramas de la animación.
- `animation-play-state`: permite pausar y reanudar la secuencia de animación.
- `animation-timing-function`: configura los tiempos de la animación; esto es, cómo la animación transiciona a través de los fotogramas mediante el establecimiento de curvas de aceleración.
- `animation-fill-mode`: configura qué valores son aplicados por la animación antes y después de que ésta es ejecutada.

Una vez que se han configurado los tiempos de la animación, es necesario definir la **apariencia** de ésta. Esto se realiza mediante el establecimiento de **dos o más fotogramas**, definidos por los keyframes. Cada fotograma describe cómo el elemento animado debe ejecutarse en un momento dado durante la secuencia de animación.

Dado que los tiempos de la animación son definidos en el estilo CSS que configura la animación, keyframes utiliza un **porcentaje** para indicar el momento de la secuencia de animación en el que tienen lugar. **0%** indica el primer momento de la secuencia de animación, mientras que **100%** indica el estado final de ésta. Debido a que estos dos momentos son tan importantes, tienen dos alias especiales: `from` y `to`. Ambos son opcionales. Si `from/0%` o `to/100%` no son especificados, el navegador inicia o finaliza la animación utilizando los valores calculados de todos los atributos.

Se pueden incluir opcionalmente **fotogramas adicionales** que describen los pasos intermedios desde el punto de partida hasta el punto final de la animación.

La sintaxis de la regla keyframe es la siguiente:

```
@keyframes <identifier> {  
  [ [ from | to | <percentage> ] [, from | to | <percentage> ]* block ]*  
}
```

Este sencillo ejemplo da estilo al elemento `<h1>` para que el texto se deslice desde el borde derecho de la ventana del navegador.

```
h1 {  
  animation-duration: 3s;  
  animation-name: slidein;  
}  
  
@keyframes slidein {  
  from {  
    margin-left: 100%;  
    width: 300%;  
  }  
  
  to {  
    margin-left: 0%;  
    width: 100%;  
  }  
}
```

El estilo dado al elemento `<h1>`, en este caso, especifica que la animación debe esperar 3 segundos para ejecutarse de principio a fin, utilizando la propiedad `animation-duration`, y que el valor de los `keyframes` que definen los fotogramas para la secuencia de animación es `slidein`.

Si quisiésemos dar un estilo personalizado al elemento `<h1>` para que apareciera en navegadores que no soportan animaciones CSS, deberíamos incluirlo aquí también. Sin embargo, en este caso, no queremos ningún estilo personalizado además del efecto de animación.

Los fotogramas son definidos utilizando los `keyframes`. En este caso, solamente tenemos dos fotogramas. El primero se ejecuta en el 0% (utilizando el alias `from`). Además, aquí configuramos que el margen izquierdo del elemento se ejecute al 100% (esto es, en el extremos derecho del elemento contenedor), y que la anchura del elemento sea 300% (o tres veces la anchura del elemento contenedor). Esto provoca que el primer fotograma de la animación tenga el encabezado fijado en el borde derecho de la ventana del navegador.

El segundo (y final) fotograma se ejecuta al 100% (utilizando el alias `to`). El margen izquierdo se establece en 0% y el ancho del elemento en 100%. Esto hace que el encabezado finalice su animación a ras del borde izquierdo del área de contenido.

Añadamos ahora otro fotograma al ejemplo de animación anterior. Digamos que queremos que la fuente del encabezado aumente a medida que se mueve de derecha a izquierda por un tiempo, y después disminuya de nuevo hasta su tamaño original. Es tan sencillo como añadir este fotograma:

```
75% {  
    font-size: 300%;  
    margin-left: 25%;  
    width: 150%;  
}
```

Esto indica al navegador que al 75% del "camino" de la secuencia de animación, la cabecera debe tener su margen izquierdo al 25% y la anchura al 150%.

Para hacer que la animación se repita por sí sola, simplemente hay que utilizar la propiedad `animation-iteration-count` para indicar cuántas veces tiene ésta que repetirse. En este caso, utilizaremos `infinite` para que la animación se repita de forma indefinida:

```
h1 {  
  animation-duration: 3s;  
  animation-name: slidein;  
  animation-iteration-count: infinite;  
}
```

El ejemplo anterior hacía que la animación se repitiera, pero el hecho de que la animación salte de nuevo al comienzo es algo que resulta extraño. Lo que realmente queremos es que se mueva hacia atrás y adelante de la pantalla. Esto se logra fácilmente estableciendo la propiedad `animation-direction` con el valor `alternate`:

```
h1 {  
  animation-duration: 3s;  
  animation-name: slidein;  
  animation-iteration-count: infinite;  
  animation-direction: alternate;  
}
```

Se puede obtener un control adicional sobre las animaciones - así como información útil sobre éstas -, mediante el uso de eventos de animación. Estos eventos, representados por el objeto `AnimationEvent`, pueden ser utilizados para detectar cuándo comienzan las animaciones, terminan o inician una nueva iteración. Cada evento incluye el momento en el que se produjo, así como el nombre de la animación que desencadenó el evento.

Nosotros modificaremos el texto desizante del ejemplo anterior para emitir información sobre cada evento de animación cuando se produce y así poder echar un vistazo a cómo funcionan.

Usaremos código JavaScript para "escuchar" los tres posibles eventos de animación. La función `setup()` configura los detectores de eventos:

```
function setup() {  
    var e = document.getElementById("watchme");  
    e.addEventListener("animationstart", listener, false);  
    e.addEventListener("animationend", listener, false);  
    e.addEventListener("animationiteration", listener, false);  
  
    var e = document.getElementById("watchme");  
    e.className = "slidein";  
}
```

Este es un código bastante estándar; se puede obtener más información sobre cómo funciona en la documentación de `element.addEventListener()`. Lo último que la función `setup()` realiza aquí es establecer la `class` sobre el elemento que estamos animando a `slidein`; hacemos esto para iniciar la animación.

¿Por qué? Porque el evento `animationstart` se ejecuta en cuanto la animación se inicia, y en nuestro caso, esto sucede antes de ejecutar el código. Así que iniciaremos la animación nosotros mismos.

Los eventos se "entregan" con la función `listener()`, que se muestra a continuación:

```
function listener(e) {  
    var l = document.createElement("li");  
    switch(e.type) {  
        case "animationstart":  
            l.innerHTML = "Started: elapsed time is " + e.elapsedTime;  
            break;  
        case "animationend":  
            l.innerHTML = "Ended: elapsed time is " + e.elapsedTime;  
            break;  
        case "animationiteration":  
            l.innerHTML = "New loop started at time " + e.elapsedTime;  
            break;  
    }  
    document.getElementById("output").appendChild(l);  
}
```

Este código también es muy sencillo. Simplemente "mira" al `event.type` para determinar qué tipo de evento de animación se produce, y a continuación añade una nota apropiada al `` (lista no ordenada) que estamos usando para registrar estos eventos.

La salida, cuando todo está dicho y hecho, se vería algo así:

- Iniciado: el tiempo transcurrido es 0
- Nuevo ciclo comenzó en el momento 3.01200008392334
- Nuevo ciclo comenzó en el momento 6.00600004196167
- Finalizado: tiempo transcurrido es 9.234000205993652

Hay que tener en cuenta que los tiempos están muy cerca entre sí, y que no se muestran exactamente igual que los establecidos cuando la animación fue configurada. También hay que saber que después de la iteración final de la animación, el evento `animationiteration` no se envía; en su lugar se envía el evento `animationend`.

Ejercicio 3

[Ver enunciado \(#ej03\)](#)

Esta página se ha dejado vacía a propósito

Capítulo 10

La especificación oficial y el estado actual de desarrollo del módulo *Transforms* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-transforms/>.

Al modificar el espacio de coordenadas, las transformaciones CSS permiten **cambiar la posición del contenido** afectado sin interrumpir el flujo normal del resto de cajas. Se llevan a cabo mediante un conjunto de propiedades CSS que permiten aplicar transformaciones lineales afines a los elementos HTML. Estas transformaciones incluyen la **rotación, inclinación, escala y traslación** tanto en el plano como en un espacio tridimensional.

Se utilizan **dos propiedades** principales para definir las transformaciones CSS: `transform` y `transform-origin`.

- `transform-origin`: especifica la posición de origen de la transformación. Por defecto se encuentra en el centro del elemento (como si definiésemos el punto en 50% 50%), pero se puede definir cualquier otro punto. Es utilizado por varias transformaciones, como rotación, escala o inclinación, que necesitan un punto inicial como parámetro.
- `transform`: especifica las transformaciones a aplicar al elemento. Se trata de una lista de funciones de transformación separadas por espacios, que se aplican una detrás de otra.

Las siguientes propiedades adicionales, añaden control adicional a las transformaciones, permitiendo incluso realizar transformaciones 3D:

- `perspective`: permite cambiar la perspectiva de los elementos y transmitir la sensación de encontrarse en un entorno en tres dimensiones.
- `perspective-origin`: especifica la posición de origen de la perspectiva.
- `transform-style`: permite a los elementos transformados en 3D y a sus descendientes también transformados en 3D, compartir un espacio 3D común.

A continuación, se muestra el listado completo de funciones de transformación 2D disponibles en CSS 3:

- `translate(<translation-value>[, <translation-value>])`: aplica una traslación 2D especificada por el vector `[tx, ty]`. Si no se indica el valor `ty`, su valor es 0.
 - `translateX(<translation-value>)`: aplica una traslación 2D en el eje x.
 - `translateY(<translation-value>)`: aplica una traslación 2D en el eje y.
- `scale(<number>[, <number>])`: aplica una operación de escalado del elementos especificada por el vector `[sx, sy]`. Si no se indica el valor `sy`, su valor es igual a `sx`.
 - `scaleX(<number>)`: aplica una operación de escalado utilizando el vector de escalado `[sx, 1]`.
 - `scaleY(<number>)`: aplica una operación de escalado utilizando el vector de escalado `[1, sy]`.
- `rotate(<angle>)`: aplica una operación de rotación especificada por los ángulos del parámetro, y tomando como origen de la roptación el punto definido por `transform-origin`.
- `skew(<angle>[, <angle>])`: aplica una operación de inclinación especificada por el vector `[ax, ay]`. Si no se indica el valor de `ay`, su valor es 0.

- `skewX(<angle>)`: aplica una operación de inclinación en el eje x.
- `skewY(<angle>)`: aplica una operación de inclinación en el eje y.

En este ejemplo se crea un `iframe` que permite utilizar la página principal de Google, rotada 90 grados desde su esquina inferior izquierda.

```
div {  
    transform: rotate(90deg);  
    transform-origin: bottom left;  
}  
  
<div>  
    <iframe src="http://www.google.com/" width="500"  
    height="600"></iframe>  
</div>
```

En este ejemplo se crea un `iframe` que permite utiliza la página principal de Google, inclinada 10 grados y trasladada 150 pixels en el eje X.

```
div {  
    transform: skewx(10deg) translateX(150px);  
    transform-origin: bottom left;  
}  
  
<div>  
    <iframe src="http://www.google.com/" width="600"  
    height="500"></iframe>  
</div>
```

La realización de transformaciones CSS en el espacio es algo más complejo. Hay que empezar configurando el espacio 3D, dándole un punto de vista. Después, hay que configurar cómo se comportarán los elementos 2D en ese espacio tridimensional.

El primer elemento a configurar es la perspectiva. La perspectiva es lo que da la impresión de tres dimensiones. Cuánto más lejos del espectador se encuentran los elementos, más pequeños se nos muestran.

Cuán rápido estos elementos reducen su tamaño es definido por la propiedad `perspective`. Cuanto más pequeño es su valor, más profunda es la perspectiva.

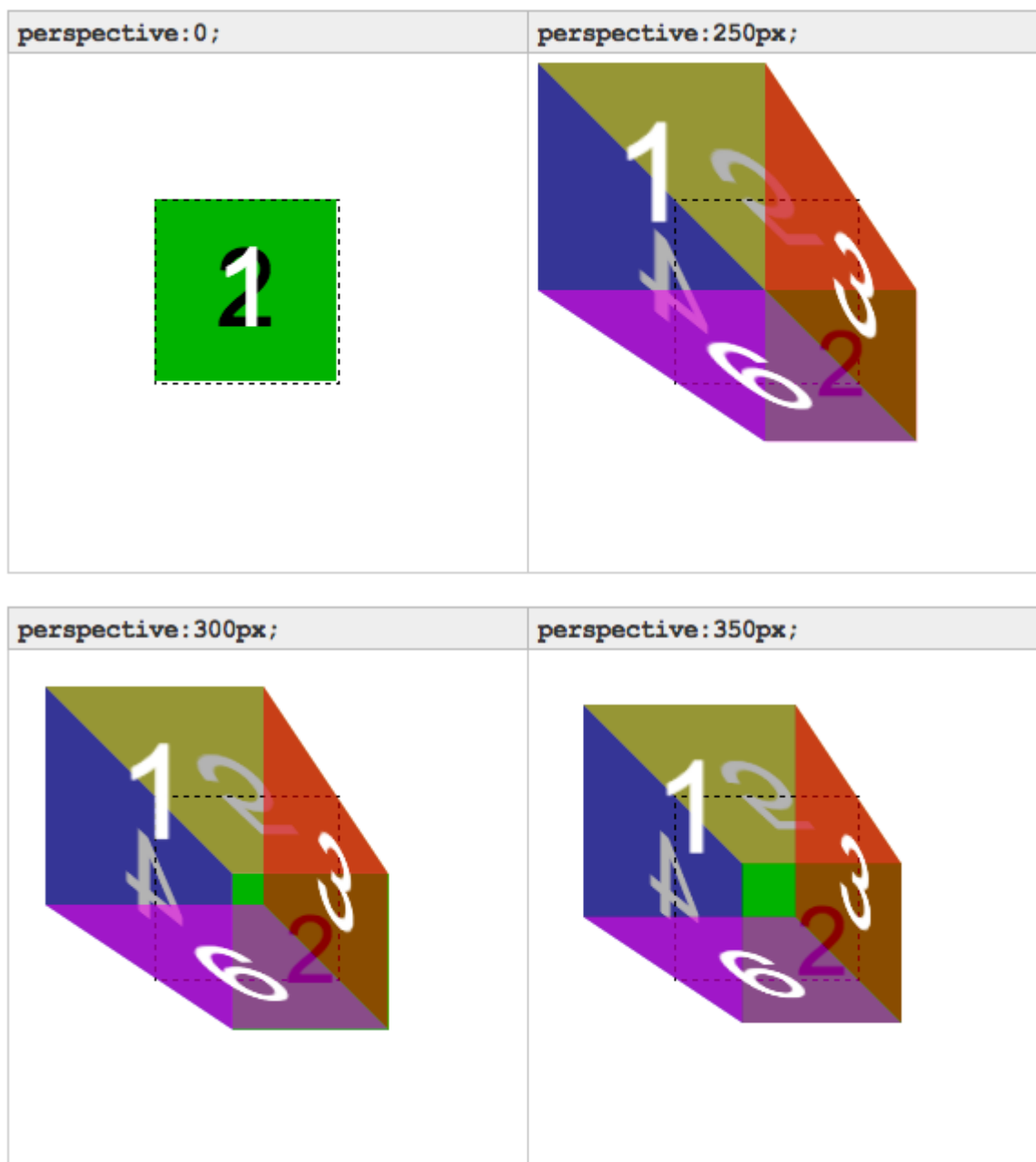


Figura 10.1 Profundidad de la perspectiva

El segundo elemento a configurar es la posición del espectador, con la propiedad `perspective-origin`. Por defecto, la perspectiva se centra en el espectador, lo cual no siempre es lo adecuado.

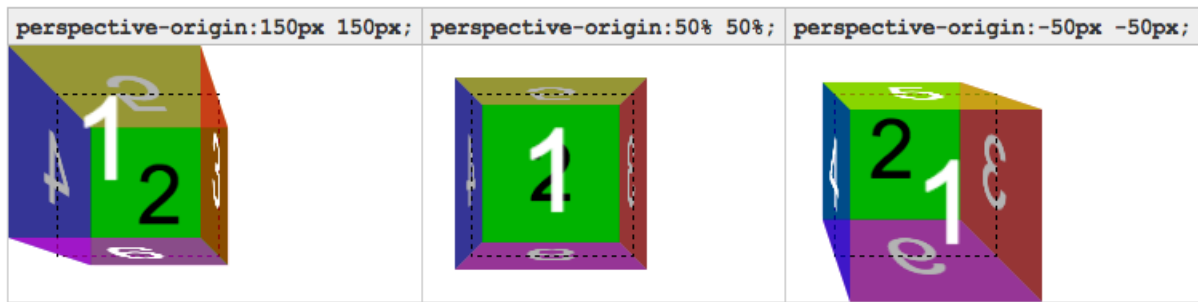


Figura 10.2 Posición del espectador

Una vez realizado esto, se puede trabajar sobre el elemento en el espacio 3D.

Ejercicio 4

[Ver enunciado \(#ej04\)](#)

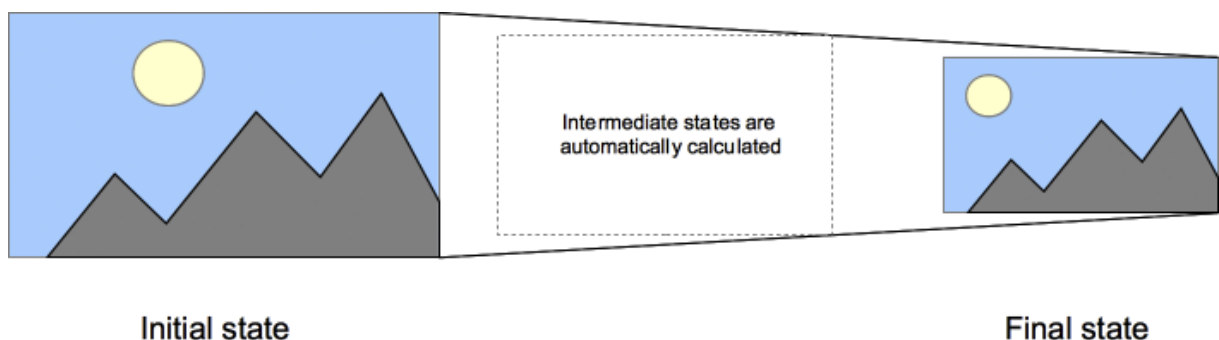
Esta página se ha dejado vacía a propósito

Capítulo 11

La especificación oficial y el estado actual de desarrollo del módulo *Transitions* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-transitions/>.

Las **transiciones CSS**, que forman parte del conjunto de especificaciones de **CSS3**, proporcionan una forma de controlar la velocidad de la animación al cambiar las propiedades CSS. Los cambios en las propiedades no tienen efecto inmediato, sino que se puede establecer un periodo de tiempo para que éstos se ejecuten. Por ejemplo, al cambiar el color de un elemento de blanco a negro, este cambio normalmente se ejecuta de manera inmediata. Sin embargo, con las transiciones CSS, los cambios se producen con intervalos de tiempo que siguen una curva de aceleración, y pueden ser personalizados.

Las animaciones que implican transición entre dos estados se llaman a menudo *transiciones implícitas*, ya que los estados intermedios entre el inicial y el final son implícitamente definidos por el navegador.



Las transiciones CSS permiten decidir qué propiedades animar (mediante su inclusión explícita), cuándo comenzará esta animación (estableciendo un retraso o *delay*), cuánto durará (estableciendo una duración), y cómo se ejecutará (definiendo una función de tiempo).

Se puede definir **qué propiedad debe ser animada y en qué manera**. Esto permite la creación de transiciones complejas. Como la animación de algunas propiedades no tiene sentido, la lista de propiedades (<http://www.w3.org/TR/css3-transitions/#animatable-types>) "animables" se limita a un conjunto finito.

También el valor `auto` es un caso complejo. La especificación nos dice que no debemos animar desde y hacia dicho valor. Algunos agentes de usuario, como los basados en Gecko o WebKit, implementan este requisito, que al usar animaciones con `auto` nos puede dar lugar a resultados impredecibles, dependiendo del navegador y su versión; por lo que debemos evitarlo.

También se debería tener cuidado al utilizar una transición inmediatamente después de añadir el elemento al DOM utilizando `.appendChild()` o borrando su propiedad `display: none;`. Esto se nos muestra como si el estado inicial nunca hubiese existido y el elemento estuviese siempre en su estado final. La manera más sencilla de superar esta limitación es aplicar un `window.setTimeout()` de algunos milisegundos antes de cambiar la propiedad CSS a la que se pretende aplicar la transición.

El primer paso al crear una transición, es especificar la propiedad o propiedades sobre las que se va a aplicar los efectos de la transición. Podemos decidir que sean todas las propiedades, ninguna o un listado de ellas.

```
transition-property: none | all | [ <property> ] [, <property> ]*
```

```
transition-property: all;
```

```
transition-property: none;
```

```
transition-property: background-color;
```

```
transition-property: background-color, height, width;
```


Si se indica la palabra reservada `all`, todas las propiedades que sean capaces de ser animadas y para las que se ha definido un cambio, serán animadas. Si se especifica `none`, ninguna propiedad será animada.

La propiedad `transition-duration` acepta una lista separada por comas de tiempos, especificadas en segundos o milisegundos, que determinan cuánto tiempo tarda cada propiedad, en completar la transición.

```
transition-duration: <time> [, <time>]*  
  
transition-duration: 2s;  
transition-duration: 4000ms;  
transition-duration: 4000ms, 8000ms;
```

La propiedad `transition-timing-function` es utilizada para especificar el ritmo en el que se producen los cambios durante la transición. Esto puede realizarse de dos maneras: indicando el nombre de una *función de tiempo* (`ease`, `linear`, `ease-in`, `ease-out` o `ease-in-out`) o definiendo una función de tiempo personalizada (especificando cuatro coordenadas para definir una curva bezier).

```
transition-timing-function: <timing-function> [, <timing-function>]*  
  
transition-timing-function: ease;  
transition-timing-function: ease, linear;  
transition-timing-function: cubic-bezier(0.6, 0.1, 0.15, 0.8);
```

Vemos lo primero de todas las funciones predefinidas. Este ejemplo muestra cómo se comporta cada una de las funciones. Cada caja se expandirá de 150px a 300px, con una duración de la transición de 3 segundos.

`ease`
`linear`
`ease-in`
`ease-out`
`ease-in-out`

Además de estas funciones de tiempo predefinidas, tenemos la posibilidad de declarar nuestra propia función de tiempo, utilizando una función cubic-bezier.

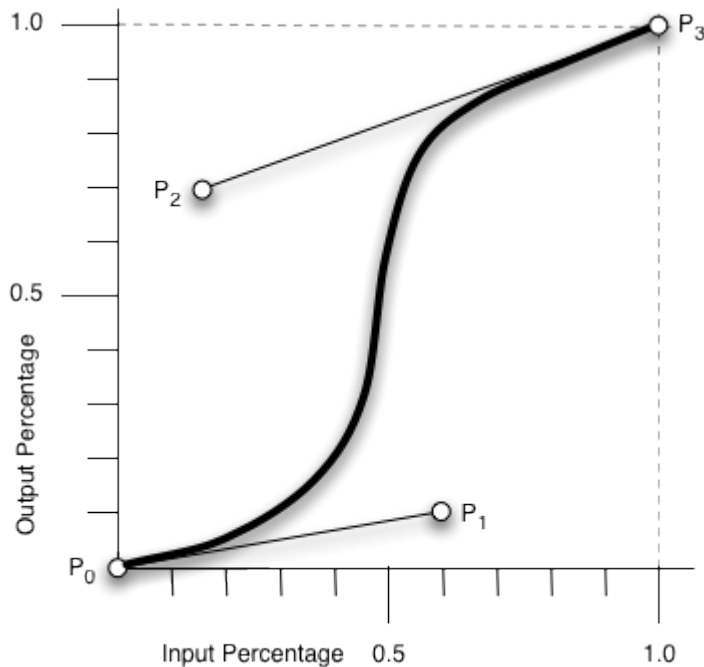


Figura 11.1 Puntos de control en una curva bezier

Si no se especifica ninguna función de tiempo, por defecto se aplica `ease`.

El último paso para crear una transición, es especificar un retraso (opcional) en el inicio de la transición. Aquí también podemos especificar una lista de tiempos, en segundos o milisegundos, que determinan el inicio de la transición desde que esta se lanza. El valor por defecto es `0`, esto es, se inicia de inmediato.

En este caso los valores negativos sí son aceptados. En este caso, la transición se iniciará tan pronto sea posible, pero dará la impresión que ya lleva tiempo ejecutándose.

```
transition-delay: <time> [, <time>]*
```

```
transition-delay: 5s;
```

```
transition-delay: 4000ms, 8000ms;
```

```
transition-delay: -5s;
```

-2s delay

no delay
1s delay
2s delay
3s delay
5s delay

Como de costumbre, disponemos de la propiedad *shorthand* que nos permite definir todas las propiedades de una sola vez.

```
transition: <transition> [, <transition>]*  
  
<transition> = <transition-property> <transition-duration>  
                <transition-timing-function> <transition-delay>  
  
transition: background-color 3s linear 1s;  
transition: 4s ease-in-out;  
transition: 5s;
```

El único valor requerido en esta propiedad es transition-duration.

Si la lista de valores de cualquier propiedad es más corta que otras, sus valores son repetidos hasta hacer que coincidan. Por ejemplo:

```
div {  
    transition-property: opacity, left, top, height;  
    transition-duration: 3s, 5s;  
}
```

Esto es tratado como si fuese:

```
div {  
    transition-property: opacity, left, top, height;  
    transition-duration: 3s, 5s, 3s, 5s;  
}
```

De manera similar, si la lista de valores de cualquier propiedad es más larga que la de transition-property, es acortado, por lo que si tienes este código CSS:

```
div {  
    transition-property: opacity, left;
```

```
    transition-duration: 3s, 5s, 2s, 1s;
}
```

Se interpreta como:

```
div {
    transition-property: opacity, left;
    transition-duration: 3s, 5s;
}
```

Existe un sólo evento que se dispara cuando las transiciones se completan. En todos los navegadores que cumplen el estándar, el evento es `transitionend`, en WebKit es `webkitTransitionEnd`. El evento `transitionend` ofrece dos propiedades:

- `propertyName`: *string* que indica el nombre de la propiedad CSS cuya transición está completada.
- `elapsedTime`: un *float* que indica el número de segundos que la transición ha estado ejecutándose en el momento en el que se dispara el evento. Este valor no está afectado por el valor de `transition-delay`.

Como de costumbre, se puede utilizar el método `element.addEventListener()` para supervisar este evento:

```
el.addEventListener("transitionend", updateTransition, true);
```

Las transiciones con una buena herramienta para crear una apariencia mucho más equilibrada sin tener que modificar la funcionalidad JavaScript. Por ejemplo:

```
<p>Click anywhere to move the ball</p>
<div id="foo"></div>
```

Utilizando JavaScript:

```
var f = document.getElementById('foo');
document.addEventListener('click', function(ev){
    f.style.left = (ev.clientX-25)+'px';
    f.style.top = (ev.clientY-25)+'px';
}, false);
```

Con CSS simplemente es necesario añadir una transición al elemento:

```
p {  
    padding-left: 60px;  
  
#foo {  
    border-radius: 50px;  
    width: 50px;  
    height: 50px;  
    background: #c00;  
    position: absolute;  
    top: 0;  
    left: 0;  
    transition: all 1s;  
}
```

Ejercicio 5

[Ver enunciado \(#ej05\)](#)

Esta página se ha dejado vacía a propósito

Capítulo 12

La especificación oficial y el estado actual de desarrollo del módulo *Text* en CSS 3 puede consultarse en <http://www.w3.org/TR/css3-text/>.

CSS 3 ha añadido nuevas funcionalidades relacionadas con el texto, realizando un importante avance en este sentido y facilitando la creación de efecto de texto, sin necesidad de recurrir a complejas soluciones.

Finalmente se ha añadido la posibilidad de crear sombras en los textos del documentos, eliminando la necesidad de crear estos textos en una herramienta externa (como Photoshop o Gimp), he incluyendolos posteriormente como una imagen en el documento, lo que perjudicaba directamente el contenido.

	text-shadow
Valores	none [<length>{2,3} && <color>?]#
Se aplica a	Todos los elementos.
Valor inicial	none
Descripción	Especifica una sombra para el texto.

Esta propiedad acepta una lista de valores separados por comas que serán aplicados al texto. Las diferentes sombras son aplicadas de arriba hacia abajo, es decir, la primera sombra de la lista está sobre las demás.

Los valores de la sombra son interpretados de la siguiente manera:

- El primer valor indica la separación horizontal de la sombra. Un valor positivo desplaza la sombra hacia la derecha de la caja, mientras que un valor negativo la desplazará hacia la izquierda.
- El segundo valor indica la separación vertical de la sombra. Un valor positivo desplaza la sombra debajo de la caja, mientras que un valor negativo la desplazará hacia arriba.
- El tercer valor es la difusión de la sombra. Si el valor es 0 (no se aceptan valores negativos) la sombra será muy nítida. Para valores mayores que 0, la sombra se va difuminando.

Algunos ejemplos:

```
h3 {color: white; text-shadow: 0.1em 0.1em 0.2em black; }
```

```
h3 {text-shadow: 0.2em 0.5em 0.1em #600,  
               -0.3em 0.1em 0.1em #060,  
               0.4em -0.3em 0.1em #006}
```

```
h3 {text-shadow: -1px -1px white, 1px 1px #333}  
h3 {text-shadow: 1px 1px white, -1px -1px #444}
```

Hasta la introducción de esta nueva característica en CSS 3, a la hora de escoger una fuente para ser utilizada en un documento HTML, nos encontrábamos con la limitación de que el usuario tuviera ese tipo de letra instalada en su ordenador, porque de no ser así, los textos se mostrarían con otras tipografías, distintas a las que habíamos elegido (las definidas como `fallback` o la fuente por defecto del navegador). Es por ello que el abanico de fuentes que podíamos utilizar, estaba reducido a las típicas Arial, Verdana, Times New Roman o Sans.

La regla `@font-face` permite enlazar fuentes en línea que van a ser utilizadas en páginas web. La descripción de la fuente, define la localización

física de la fuente (local o externa), así como las características de dicha fuente. Es posible declarar tantas reglas `@font-face` como fuentes sean necesarias utilizar.

Su sintaxis es la siguiente:

```
@font-face {  
  [font-family: <family-name>;]?  
  [src: [ <uri> [format(<string>#)]? | <font-face-name> ]#;]?  
  [unicode-range: <urange>#;]?  
  [font-variant: <font-variant>;]?  
  [font-feature-settings: normal|<feature-tag-value>#;]?  
  [font-stretch: <font-stretch>;]?  
  [font-weight: <weight>;]  
  [font-style: <style>;]  
}
```

Donde:

- `<family-name>`: especifica el nombre de la fuente que se utilizará.
- `<uri>`: URL para de la ubicación remota de la fuente, o nombre de la fuente en el ordenador del autor ("FontName")
- `<font-variant>`: variante de una fuente anterior.
- `<font-stretch>`: especifica la anchura de la fuente.
- `<weight>`: especifica el peso de la fuente (light, normal, bold ...)
- `<style>`: especifica el estilo de la fuente (normal, italic o oblique)

Un ejemplo sencillo:

Los usuarios que accedan con un navegador que cumpla con el estándar, verán este texto con la fuente Delicious, *y esta parte con Delicious Bold*.

Dependiendo del tipo de navegador con el que se acceda a nuestra web, éste acepta un tipo concreto de fuente. Los tipos existentes son los siguientes:

"woff"	WOFF (Web Open Font Format) (http://www.w3.org/TR/WOFF/)	.woff
--------	--	-------

"truetype"	TrueType (http://www.microsoft.com/typography/otspec/default.htm)	.ttf
"opentype"	OpenType (http://www.microsoft.com/typography/otspec/default.htm)	.ttf, .otf
"embedded-opentype"	Embedded OpenType (http://www.w3.org/Submission/2008/SUBM-EOT-20080305/)	.eot
"svg"	SVG Font (http://www.w3.org/TR/SVG/fonts.html)	.svg

Un ejemplo de utilización de varios orígenes de fuentes es el siguiente:

```
@font-face {  
  font-family: 'ITCAvantGardeStd-Bk';  
  src: url('fonts/ITCAvantGardeStd-Bold.eot') format('eot'),  
        url('fonts/ITCAvantGardeStd-Bold.woff') format('woff'),  
        url('fonts/ITCAvantGardeStd-Bold.ttf') format('truetype'),  
        url('fonts/ITCAvantGardeStd-Bold.svg') format('svg');  
  font-weight: bold;  
  font-style: normal;  
}
```

Ejercicio 9

[Ver enunciado \(#ej09\)](#)

Capítulo 13

Dado el siguiente código HTML, y utilizando las pseudo-clases `first-child`, `first-of-type`, `nth-of-type`, `nth-last-of-type`, `last-of-type` y `last-child` conseguir el siguiente resultado:

```
<html>
<head>
  <title>Ejercicio de pseudo-clases</title>
  <style type="text/css">

    </style>
</head>
<body>
  <ul>
    <li>Primero</li>
    <li>Segundo</li>
    <li>Tercero</li>
    <li>Cuarto</li>
    <li>Quinto</li>
  </ul>
</body>
</html>
```

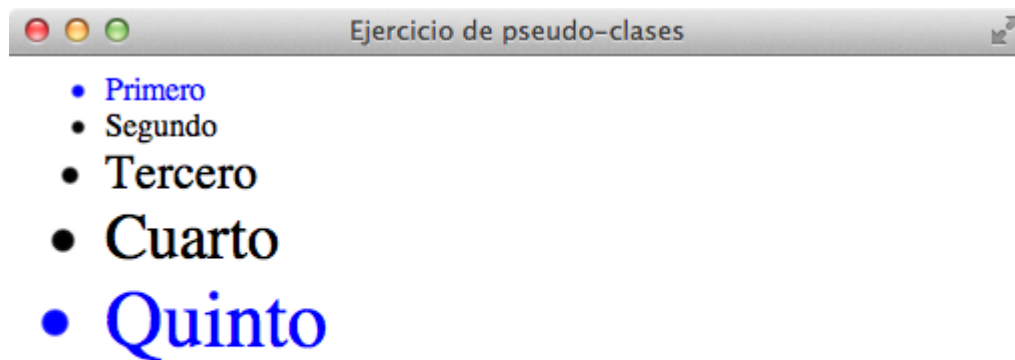


Figura 13.1 Aspecto tras aplicar los estilos utilizando las pseudo-clases

Dados el siguiente código HTML y CSS base, aplicar los *Media Query* necesarios para obtener el comportamiento mostrado en las imágenes.



Figura 13.2 Aspecto de la web a pantalla completa

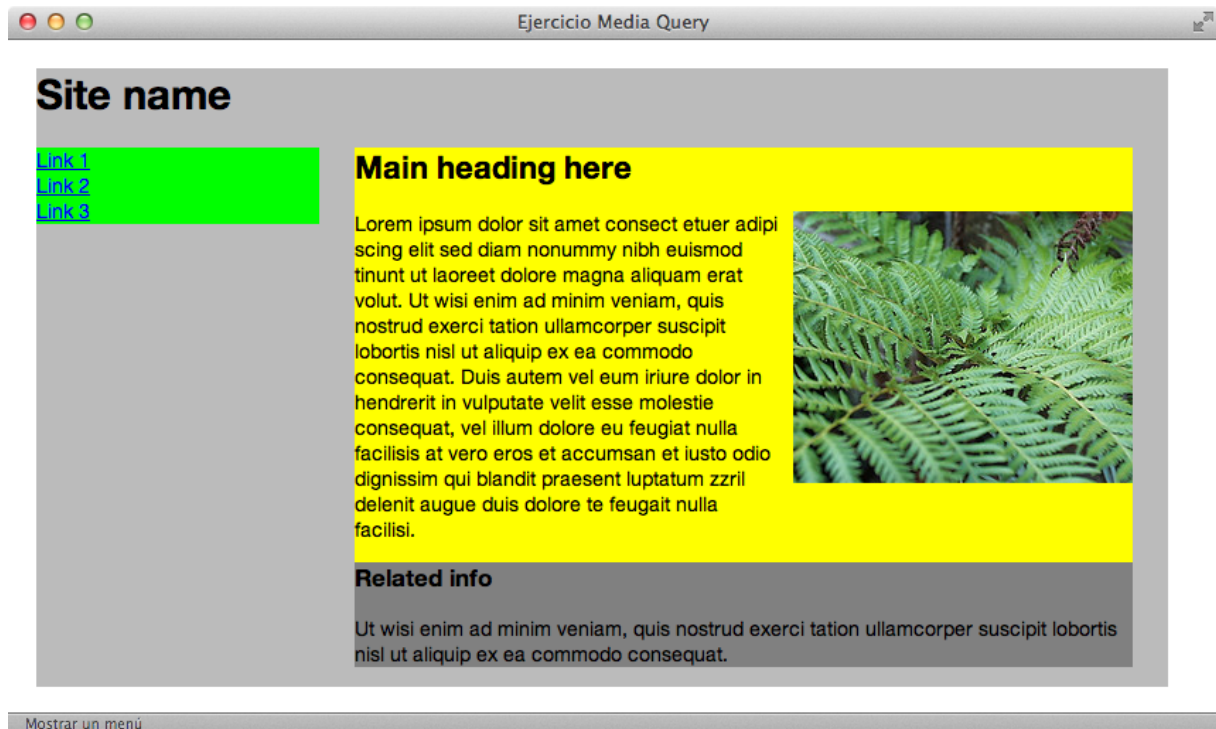


Figura 13.3 Aspecto de la web en una pantalla de 1000px

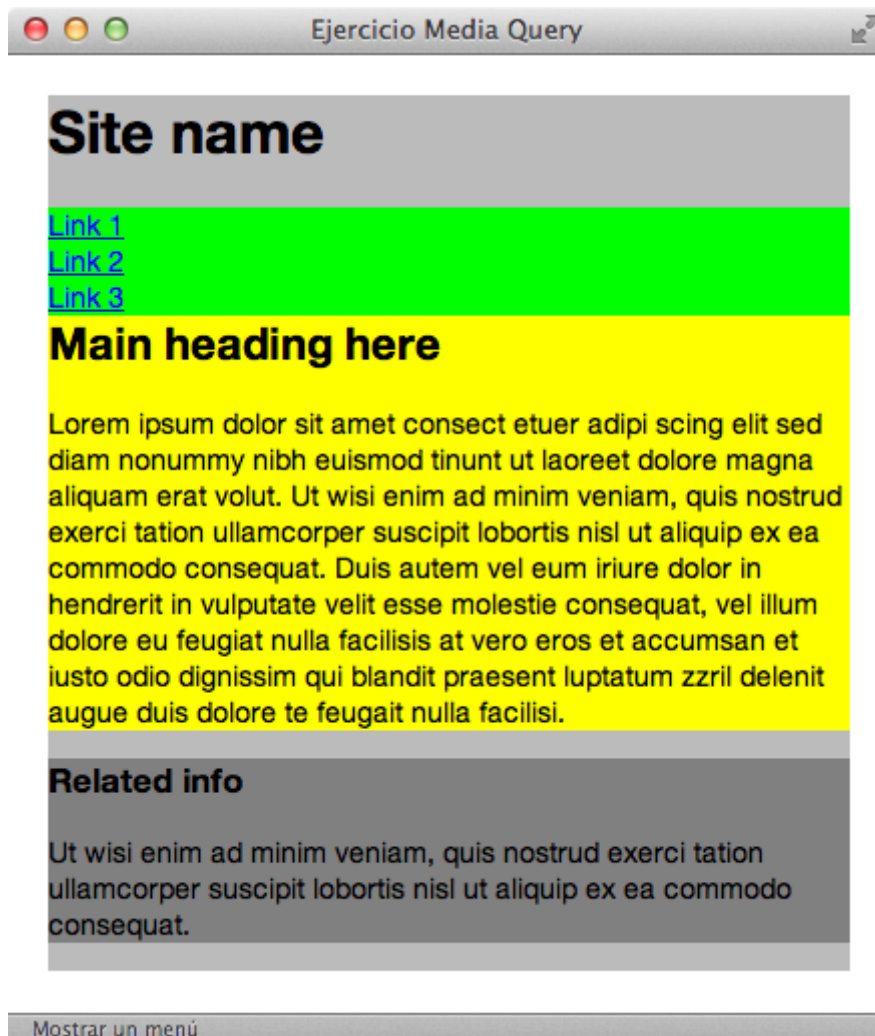


Figura 13.4 Aspecto de la web en una pantalla de 480px

Descargar los ficheros fuente ([snippets/cap12/ej02.zip](#))

Dados el siguiente código HTML y CSS base, crear las animaciones necesarias para simular un efecto de nieve. Utilizad al menos dos animaciones:

- La primera para simular la caída de los copos de nieve.
- La segunda para hacer desaparecer los copos según van llegando al suelo, utilizando la propiedad `opacity`.

La animación debe ejecutarse de manera infinita.



Figura 13.5 Copos de nieve en movimiento

Descargar los ficheros fuente ([snippets/cap12/ej03.zip](#))

Dados el siguiente código HTML y CSS base, crear y aplicar las siguientes transformaciones. Utilizar únicamente selectores específicos de CSS 3:

- Desplazar la primera caja 100px hacia la derecha, y 50px hacia abajo desde su posición original.
- Incrementar el tamaño de la segunda caja, haciéndola el doble de grande.
- Rotar la tercera caja en 45°, tanto desde su centro de coordenadas como desde la esquina superior izquierda.
- Crear un paralelogramo con la cuarta caja, cuyo ángulo de inclinación sea de 25°.

Descargar los ficheros fuente ([snippets/cap12/ej04.html](#))

Dados el siguiente código HTML y CSS base, crear las transiciones necesarias para pasar del estado inicial al final, aplicando las siguientes transformaciones:

- Rotar los elementos 270°.
- Intercambiar las posiciones de las cajas 1 y 2, 3 y 4.
- Cambiar el tamaño de 100px a 50px.
- Cambiar el color de fondo de azul a rojo.
- Cambiar el color del texto de negro a amarillo.
- Cambiar el tamaño de letra de 20px a 18px.

Utilizar en cada caso, una duración y función de tiempo diferentes.

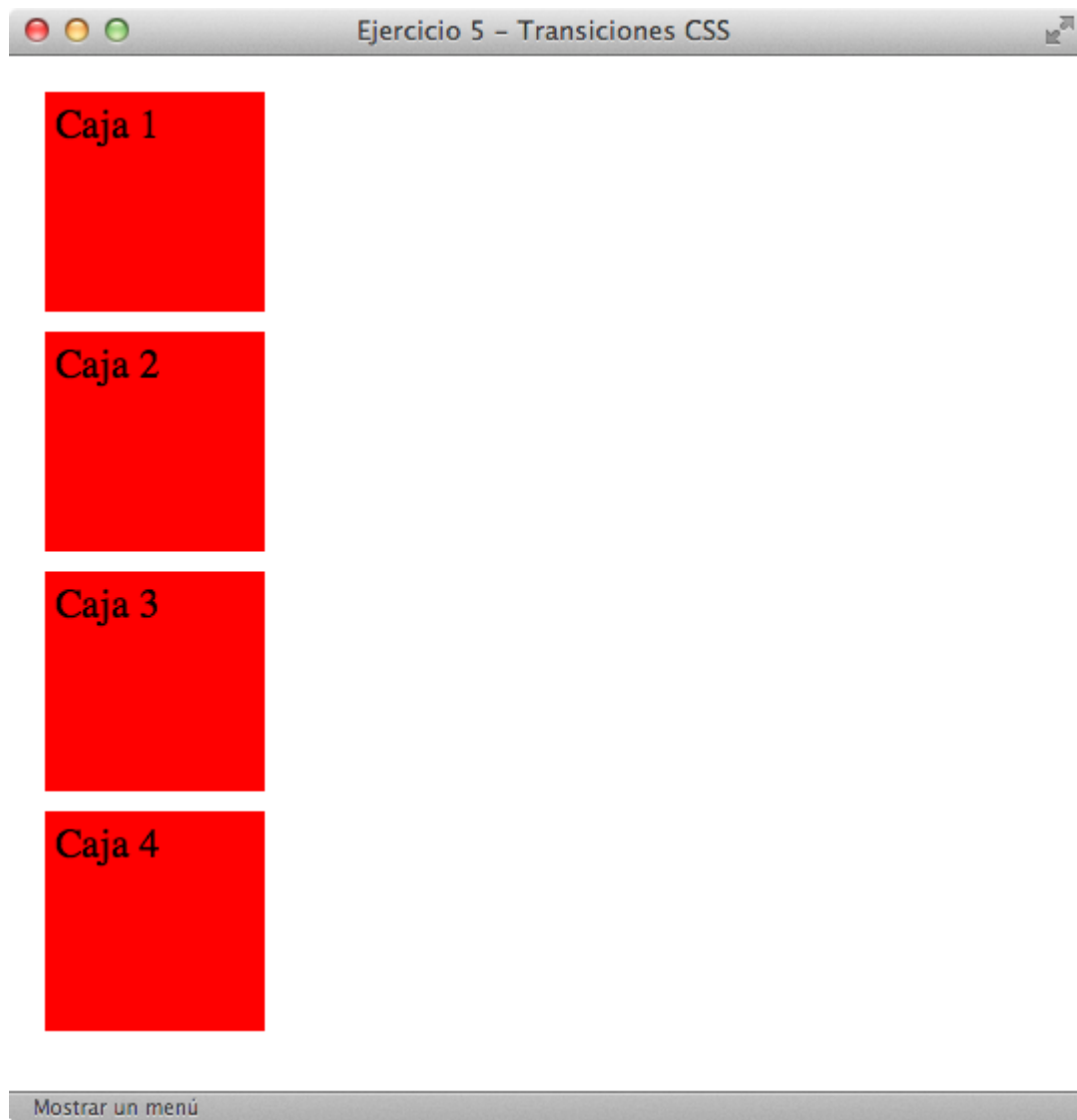


Figura 13.6 Estado inicial antes de la transición

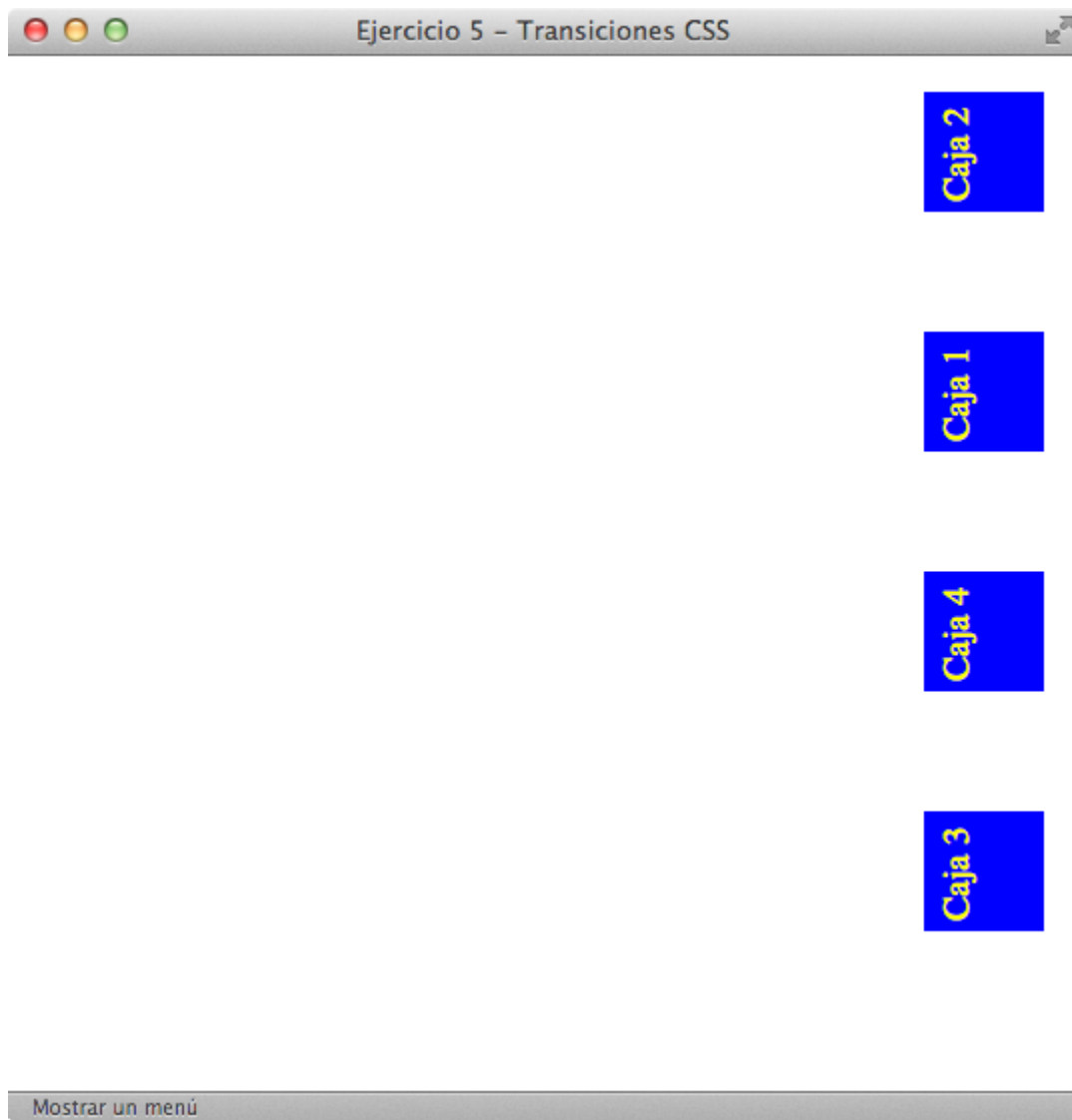


Figura 13.7 Estado final tras la transición

Descargar los ficheros fuente (<snippets/cap12/ej05.html>)

Dados el siguiente código HTML y CSS base, aplicar las reglas CSS necesarias para conseguir el siguiente aspecto:

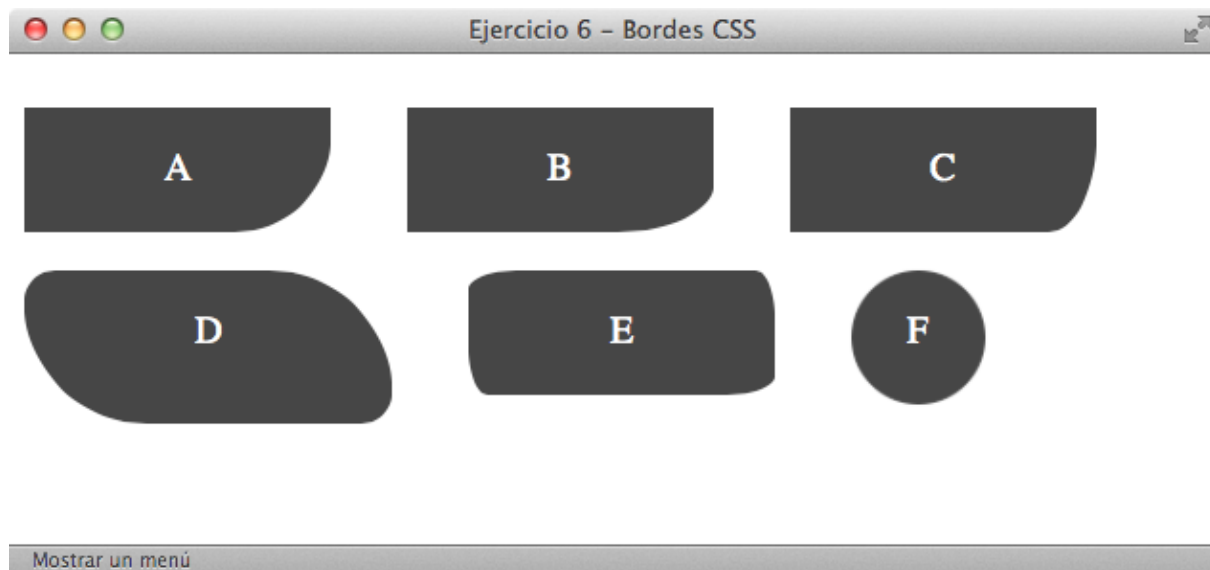


Figura 13.8 Estado final tras aplicar los estilos de borde

Descargar los ficheros fuente (<snippets/cap12/ej06.html>)

Dados el siguiente código HTML y CSS base, aplicar las reglas CSS necesarias para conseguir el siguiente aspecto:

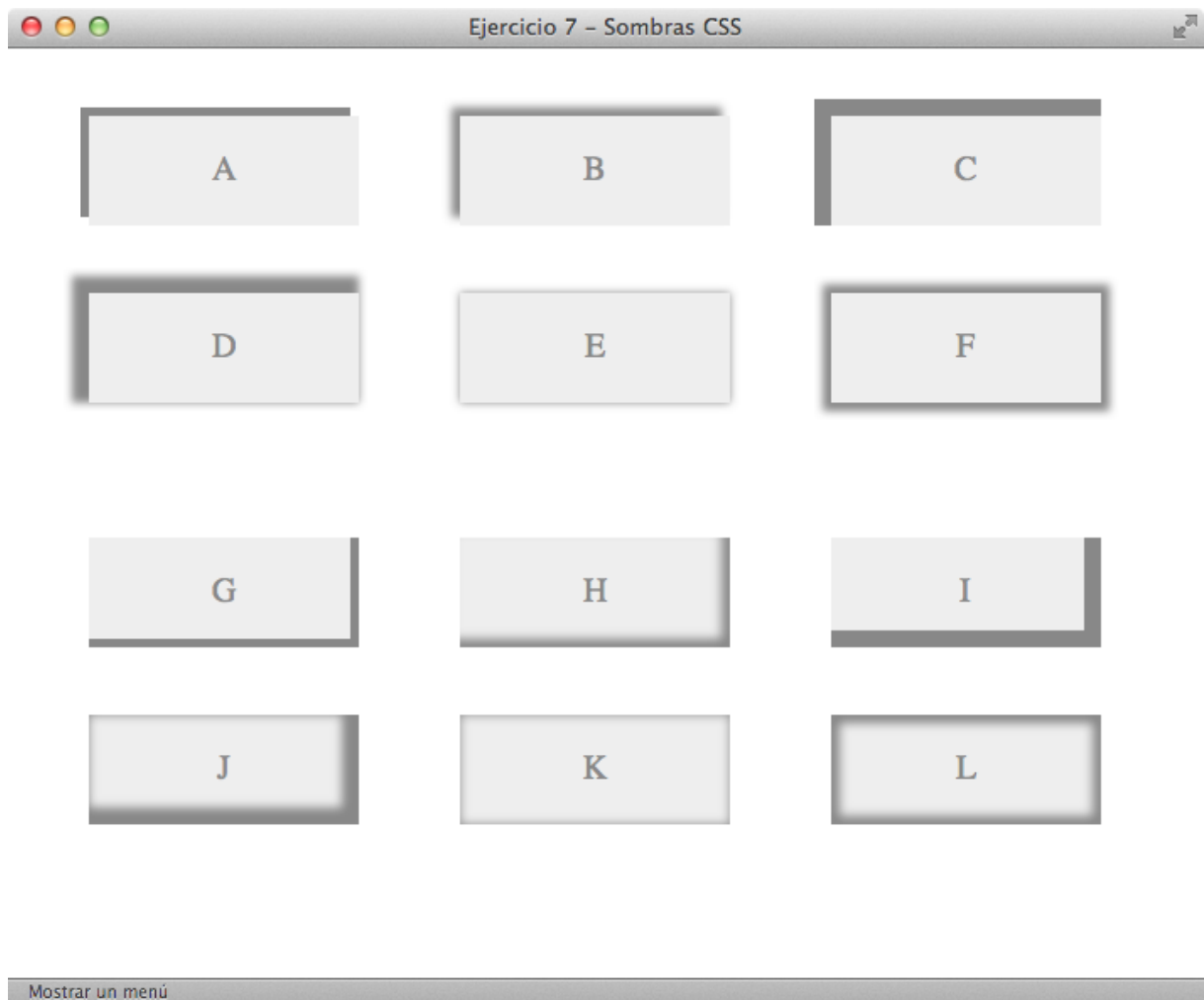


Figura 13.9 Estado final tras aplicar los estilos de sombra

Descargar los ficheros fuente (<snippets/cap12/ej07.html>)

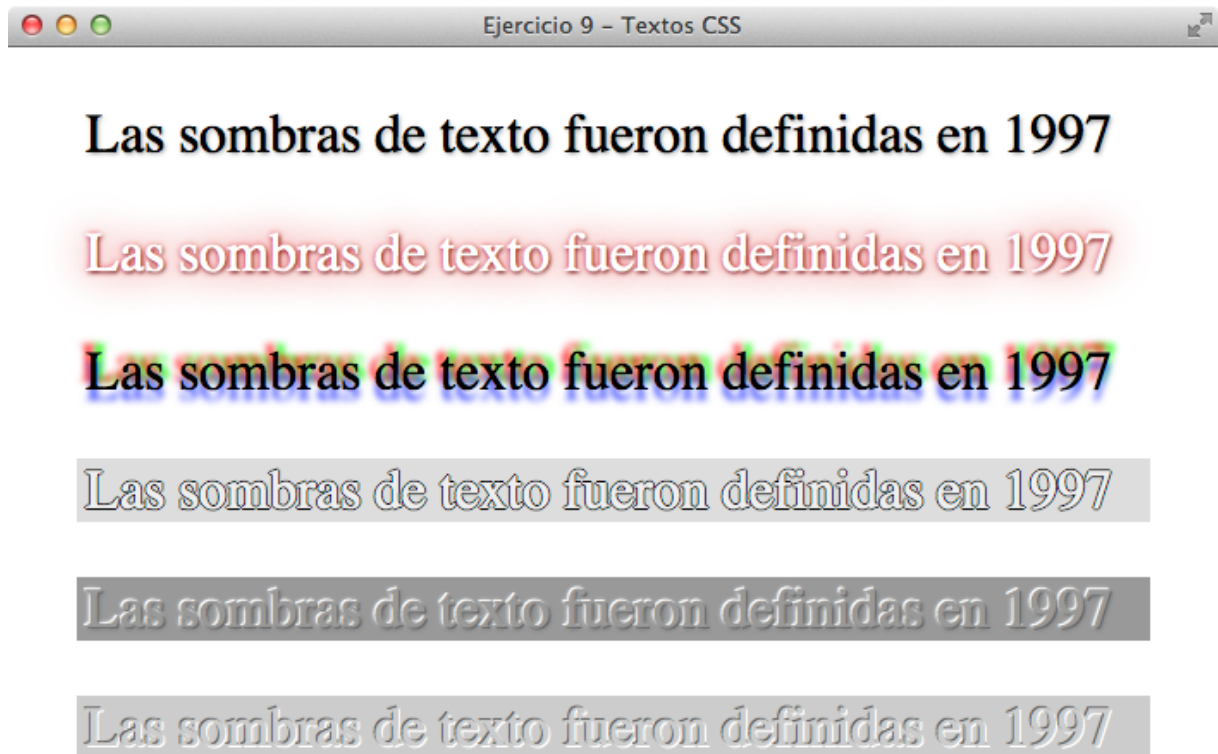
Dados el siguiente código HTML y CSS base, aplicar las reglas CSS necesarias para conseguir el siguiente aspecto:



Figura 13.10 Estado final tras definir fondos múltiples

Descargar los ficheros fuente ([snippets/cap12/ej08.zip](#))

Dados el siguiente código HTML y CSS base, aplicar las reglas CSS necesarias para conseguir el siguiente aspecto:



Mostrar un menú

Figura 13.11 Estado final tras definir fondos múltiples

Descargar los ficheros fuente (<snippets/cap12/ej09.html>)

Dados el siguiente código HTML y CSS base, aplicar las animaciones y transformaciones necesarias para conseguir el siguiente efecto:

- Al cargarse la página, únicamente se mostrarán los recuadros con las imágenes, ocultando el título, texto, enlace a "Leer más" y el fondo naranja.
- Al pasar con el ratón sobre la imagen, debe ocurrir lo siguiente:
 - La imagen de fondo aumentará su tamaño, dando la impresión de que se acerca.
 - Se mostrará el fondo naranja, de manera progresiva.
 - El título aparecerá por la parte superior de la imagen, hasta colocarse en su lugar.

- El párrafo aparecerá por la parte inferior de la imagen, hasta colocarse en su lugar.
- Se mostrará el enlace "Leer más", de manera progresiva.

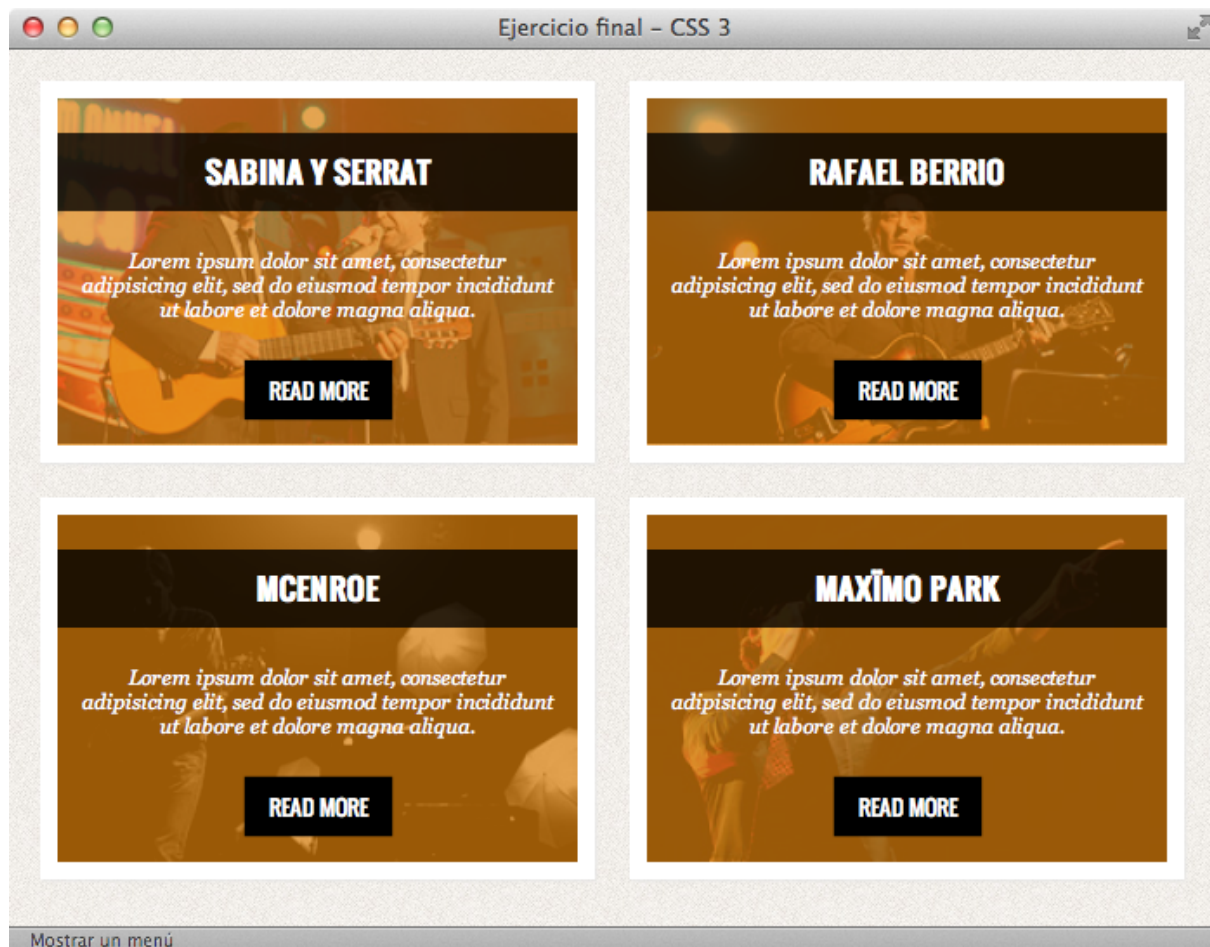


Figura 13.12 Estado final tras realizar las animaciones y transformaciones

Descargar los ficheros fuente ([snippets/final/final01.zip](#))

Dados el siguiente código HTML y CSS base, simular el comportamiento de un reloj de aguja:

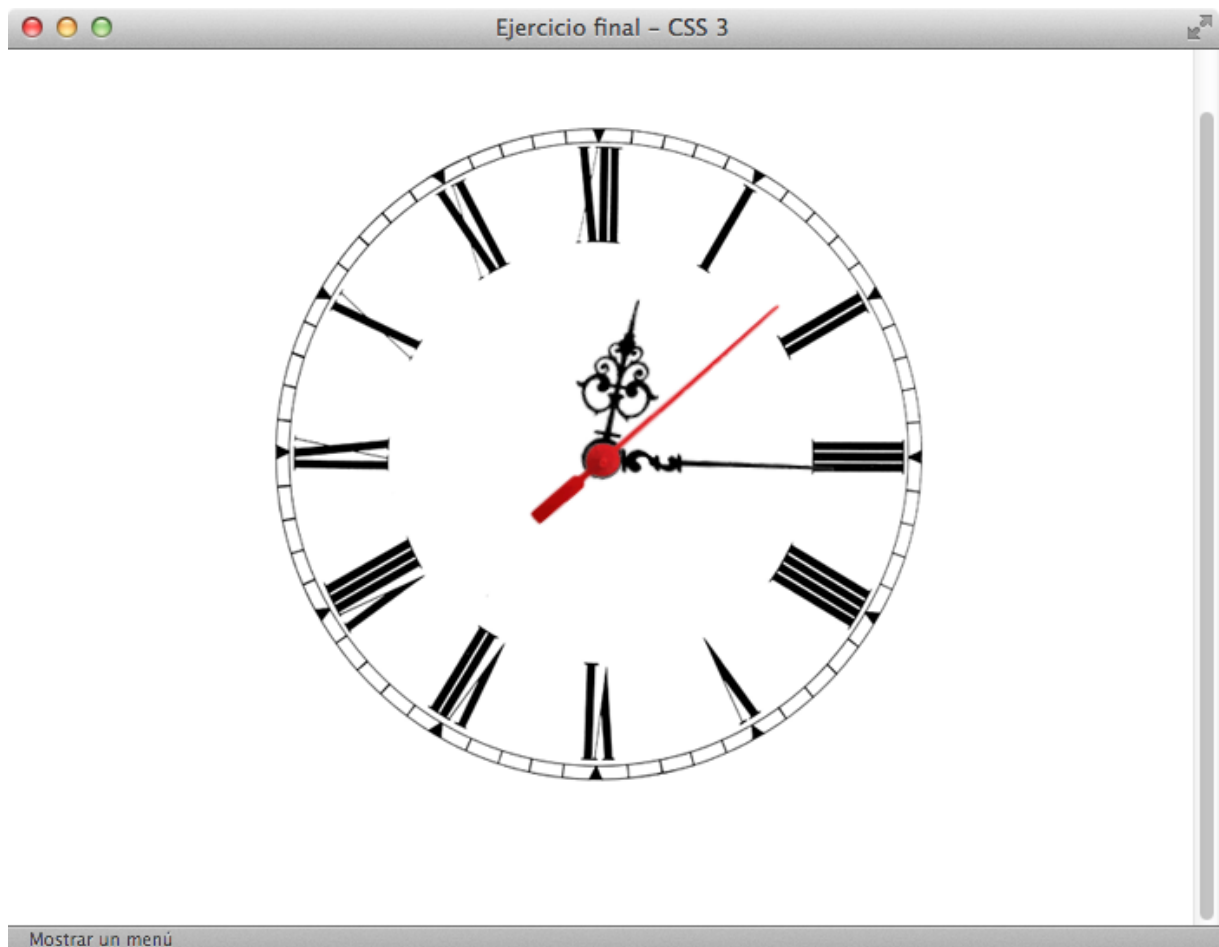


Figura 13.13 Reloj marcando las horas, minutos y segundos

Descargar los ficheros fuente ([snippets/final/final02.zip](#))