

# Introducción al diseño en CSS

El objetivo de este tutorial es conocer y practicar con las nuevas propiedades que nos presenta CSS3. Lo haremos como en el tutorial anterior, de modo que dispondremos de un concepto teórico, un ejercicio resuelto y la vista en ejecución de la página con su hoja de estilo.

## 1. Bordes

---

### 1.1. Bordes redondeados (border-radius)

La propiedad border-radius permite crear esquinas redondeadas. Especificamos en píxeles u otra medida el radio de redondeo de la o las esquinas.

Podemos indicar un único valor que se asignará a los cuatro vértices:

```
#recuadro1{  
  border-radius: 20px;  
  background-color:#ddd;  
  width:200px;  
  padding:10px;  
}
```

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

También podemos indicar el redondeo de cada vértice de forma independiente (el orden de los valores son la esquina superior izquierda, la esquina superior derecha, la esquina inferior derecha y por último la esquina inferior izquierda):

```
#recuadro2{  
  border-radius: 20px 40px 60px 80px;  
  background-color:#aa0;  
  width:200px;  
  padding:10px;  
  margin-top:10px;  
}
```

#### Recuadro 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

El texto completo sería:

```
<!DOCTYPE html>  
<html>  
<head>  
<title>Prueba</title>  
  
<style type="text/css">  
  
#recuadro1{  
  border-radius: 20px;  
  background-color:#ddd;  
  width:200px;
```

```

padding:10px;
}

#recuadro2{
border-radius: 20px 40px 40px 20px;
background-color:#aa0;
width:200px;
padding:10px;
margin-top:50px;
}

body {
background:white;
margin:50px;
}

</style>

</head>
<body>

<div id="recuadro1">
<h3>Recuadro 1</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo
ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis
parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,
</p>
</div>

<div id="recuadro2">
<h3>Recuadro 2</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo
ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis
parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,
</p>
</div>

</body>
</html>

```

## 1.2. Bordes redondeados de alguno de los vértices

Si tenemos que redondear solo alguno de los cuatro vértices podemos utilizar alguna de las siguientes propiedades:

- border-top-left-radius
- border-top-right-radius
- border-bottom-right-radius
- border-bottom-left-radius

Con esta definición de las propiedades border-top-left-radius y border-bottom-right-radius deben aparecer redondeados los vértices superior izquierdo e inferior derecho:

```
#recuadro1{
  border-top-left-radius: 20px;
  border-bottom-right-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
}
```

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

El segundo ejemplo muestra redondeado solo el vértice superior derecho:

```
#recuadro2{
  border-top-right-radius: 40px;
  background-color:#aa0;
  width:200px;
  padding:10px;
  margin-top:10px;
}
```

#### Recuadro 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

## 2. Sombras (box-shadow)

La propiedad box-shadow permite definir una sombra a un objeto de la página. Debemos definir tres valores y un color, por ejemplo:

```
#recuadro1{
  box-shadow: 30px 10px 20px #aaa;
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
}
```

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

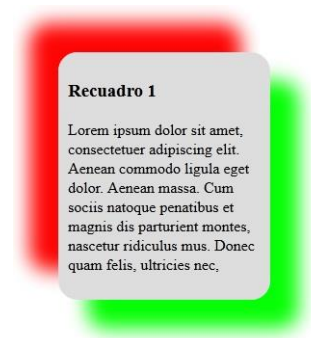
La propiedad tiene 3 valores y un color, los valores son los siguientes:

- El desplazamiento horizontal de la sombra, positivo significa que la sombra se encuentra a la derecha del objeto, un desplazamiento negativo pondrá la sombra a la izquierda.
- El desplazamiento vertical, uno negativo la sombra será en la parte superior del objeto, uno positivo la sombra estará por debajo.
- El tercer parámetro es el radio de desenfoque, si se pone a 0 la sombra será fuerte y con color liso, más grande el número, más borrosa será.
- El último valor es el color a aplicar a la sombra.

Un ejemplo completo de esta propiedad sería:



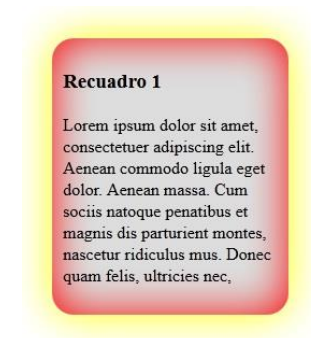
```
#recuadro1{
  box-shadow: -30px -30px 20px #f00,
              30px 30px 20px #0f0;
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
}
```



## 2.2. Sombras interiores (box-shadow)

Otra posibilidad de la propiedad box-shadow es la de implementar la sombra interior al objeto, para esto debemos anteceder a los valores la palabra inset. Por ejemplo si queremos un recuadro con sombra interior de color rojo y sombra exterior de color amarilla podemos aplicar los siguientes valores:

```
#recuadro1{
  box-shadow: inset 0px 0px 40px #f00,
              0px 0px 40px #ff0;
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
}
```



## 2.3. Sombras con transparencias (box-shadow)

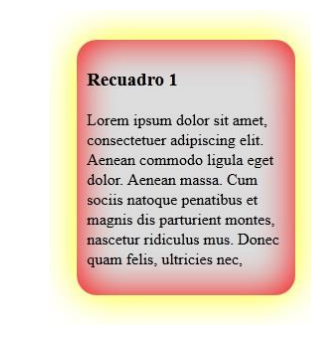
Otra posibilidad que podemos aplicar a la propiedad box-shadow es definir un valor de transparencia de la sombra. Para esto debemos utilizar la función rgba, ejemplo:

```
box-shadow: 30px 30px 30px rgba(255,0,0,0.3);
```

Mediante la función rgba indicamos el color de la sombra con los primeros tres parámetros (en este ejemplo 255 de rojo, 0 de verde y 0 de azul), luego el cuarto parámetro es el índice de transparencia (donde 0 indica que es totalmente transparente la sombra y 1 es totalmente opaca, podemos utilizar cualquier valor entre 1 y 0).

Para confeccionar un recuadro con sombra de color rojo con un índice de transparencia del 50% luego debemos especificarlo de la siguiente manera:

```
#recuadro1{
  box-shadow: 30px 30px 30px rgba(255,0,0,0.5);
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
}
```



## 2.4. Sombras de texto (text-shadow)

La propiedad text-shadow nos permite definir una sombra a un texto, la sintaxis más común es:

```
Elemento {
```

```
text-shadow: desplazamientoX desplazamientoY radio-de-  
desenfoque color;  
}
```

La propiedad tiene 4 valores que son los siguientes:

- El desplazamiento horizontal de la sombra, un desplazamiento negativo pondrá la sombra a la izquierda.
- El desplazamiento vertical, un valor negativo dispone la sombra en la parte superior del texto, uno positivo la sombra estará por debajo del texto.
- El tercer parámetro es el radio de desenfoque, si se pone a 0 la sombra será fuerte y con color liso, más grande el número, más borrosa será.
- El último parámetro es el color de la sombra.

Por ejemplo si queremos que un texto tenga una sombra en la parte inferior a derecha con un pequeño desenfoque de color gris luego debemos implementar el siguiente código:

```
#titulo1 {  
text-shadow: 5px 5px 5px #aaa;  
}
```

**Titulo sombreado**

Si queremos que la sombra se disponga en la parte superior izquierda de cada letra luego debemos definir los siguientes valores:

```
#titulo2 {  
text-shadow: -5px -5px 5px #aaa;  
}
```

**Titulo sombreado**

Otra sintaxis de text-shadow es aplicar varias sombras al texto, por ejemplo:

```
#titulo3 {  
text-shadow: 3px 3px 5px #f00,  
6px 6px 5px #0f0,  
9px 9px 5px #00f;  
}
```

**Titulo sombreado**

Nota: Es importante tener en cuenta que esta propiedad no la soporta el navegador Internet Explorer 9.

### 3. Transformaciones 2D

---

Las transformaciones todavía no están definidas como un estándar en todos los navegadores, por lo que es necesario agregar el prefijo del navegador que la implementa:

```
Elemento {  
-ms-transform: función de transformación(valor(es)); /*  
Internet Explorer */
```

```

        -webkit-transform: función de transformación(valor(es)); /*
WebKit */
        -moz-transform: función de transformación(valor(es)); /*
Firefox */
        -o-transform: función de transformación(valor(es)); /*
Opera */
    }

```

Tengamos en cuenta que la propiedad de transformación 2D definitiva será:

```

Elemento {
    transform: función de transformación(valor(es));
}

```

### 3.1. Rotación transform:rotate

La primera función de transformación que veremos será la de rotar un elemento HTML.

La función de rotación se llama rotate y tiene un parámetro que indica la cantidad de grados a rotar. La rotación es en el sentido de las agujas del reloj. Podemos indicar un valor negativo para rotar en sentido antihorario.

Para rotar un recuadro 45 grados en el sentido de las agujas de un reloj y que funcione en la mayoría de los navegadores deberemos implementar el siguiente código:

```

#recuadro1{
    -ms-transform: rotate(45deg);
    -webkit-transform: rotate(45deg);
    -moz-transform: rotate(45deg);
    -o-transform: rotate(45deg);
    transform: rotate(45deg);
    border-radius: 20px;
    background-color:#ddd;
    width:200px;
    padding:10px;
}

```



Tengamos en cuenta que se ejecuta la propiedad -ms-transform, -webkit-transform etc. según el navegador que está procesando la página, inclusive hemos agregado la propiedad transform: rotate(45deg) que será la que en un futuro todos los navegadores interpretarán.

Otra cosa importante para notar es que el punto de rotación coincide con el centro del recuadro (es como clavar un alfiler en el centro del recuadro y luego rotar el recuadro en el sentido de las agujas del reloj).

También podemos rotar en sentido antihorario indicando el valor del grado con un valor negativo:

```

#recuadro2{
    -ms-transform: rotate(-45deg);
    -webkit-transform: rotate(-45deg);
}

```

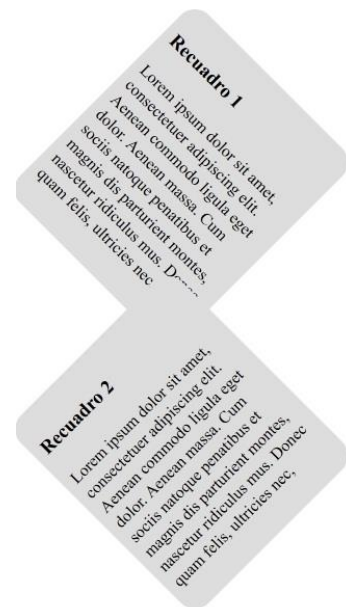


```

-moz-transform: rotate(-45deg);
-o-transform: rotate(-45deg);
transform: rotate(-45deg);
border-radius: 20px;
background-color:#ddd;
width:200px;
padding:10px;
}

```

También podemos observar que cuando la rotación se ejecuta no ocupa más espacio el elemento HTML sino que se solapa eventualmente con otros elementos de la página.



### 3.2. Punto de origen (transform-origin)

Como vimos en el punto anterior cuando rotamos un elemento HTML siempre hay un punto fijo (por defecto es el punto central del recuadro).

Podemos variar dicho punto y ubicar por ejemplo en cualquiera de los cuatro vértices del recuadro con la siguiente sintaxis:

```

Elemento {
    transform-origin: left top;
}
Elemento {
    transform-origin: right top;
}
Elemento {
    transform-origin: left bottom;
}
Elemento {
    transform-origin: right bottom;
}

```

Por ejemplo si queremos dejar fijo el vértice superior izquierdo y seguidamente rotar 10 grados en sentido horario luego debemos codificar:

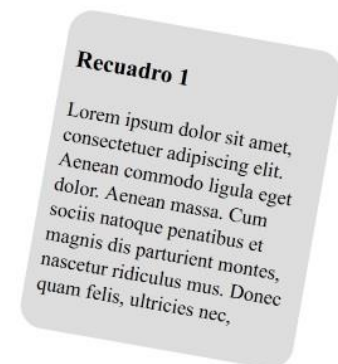
```

#recuadro1{
    -ms-transform: rotate(10deg);
    -webkit-transform: rotate(10deg);
    -moz-transform: rotate(10deg);
    -o-transform: rotate(10deg);
    transform: rotate(10deg);

    -ms-transform-origin: left top;
    -webkit-transform-origin: left top;
    -moz-transform-origin: left top;
    -o-transform-origin: left top;
    transform-origin: left top;

    border-radius: 20px;
    background-color:#ddd;
}

```





```
width:200px;
padding:10px;
}
```

Como podemos ver es como disponer un alfiler en el vértice superior izquierdo y seguidamente rotar en el sentido de las agujas de un reloj 10 grados.

Con lo visto podemos disponer el punto de origen en cinco lugares (el centro y los cuatro vértices), pero podemos trasladar a cualquier punto dentro del recuadro el punto del origen con las siguientes sintaxis:

```
transform-origin: 0% 50%;
```

El primer valor representa las "x" y el segundo valor representa las "y".

Si queremos disponer el punto de origen en el vértice superior derecho podemos hacerlo como ya conocemos:

```
Elemento {
    transform-origin: right top;
}
0 mediante porcentajes:
Elemento {
    transform-origin: 100% 0%;
}
```

Luego podemos mediante porcentajes en x e y desplazar el punto de origen a cualquier parte dentro del elemento HTML (div en este caso).

Por ejemplo si queremos disponer el punto de origen en la mitad del lado izquierdo:

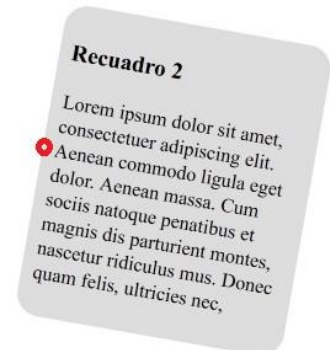
```
#recuadro2{
  -ms-transform: rotate(10deg);
  -webkit-transform: rotate(10deg);
  -moz-transform: rotate(10deg);
  -o-transform: rotate(10deg);
  transform: rotate(10deg);

  -ms-transform-origin: 0% 50%;
  -webkit-transform-origin: 0% 50%;
  -moz-transform-origin: 0% 50%;
  -o-transform-origin: 0% 50%;
  transform-origin: 0% 50%;

  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
  margin-top:50px;
}
```

El punto rojo indica el "punto de origen".

Otra posibilidad es indicar el punto con alguna medida permitida en CSS (px, em etc.), por ejemplo podemos indicar que el punto de origen este 20 píxeles en x y 40 en y:



```
#recuadro3{
  -ms-transform: rotate(10deg);
  -webkit-transform: rotate(10deg);
  -moz-transform: rotate(10deg);
  -o-transform: rotate(10deg);
  transform: rotate(10deg);

  -ms-transform-origin: 20px 40px;
  -webkit-transform-origin: 20px 40px;
  -moz-transform-origin: 20px 40px;
  -o-transform-origin: 20px 40px;
  transform-origin: 20px 40px;

  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
  margin-top:50px;
}
```

### Recuadro 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

### 3.3. Escalado (transform:scale)

Otra función de transformación 2D es el escalado, esta función permite agrandar o reducir el tamaño del elemento. La primera sintaxis que podemos utilizar es la siguiente:

```
Elemento {
  transform: scale(valorx ,valory);
}
```

El primer parámetro indica la escala para x (con el valor 1 el elemento queda como está, con un valor mayor crece y con un valor menor decrece), el segundo parámetro es para la escala en y.

Por ejemplo si queremos que sea del doble de ancho y la mitad de altura sería:

```
Elemento {
  transform: scale(2 ,0.5);
}
```

Un recuadro escalado con 20% menos de ancho y 20% más de alto (definiendo el punto de origen en el vértice superior izquierdo:

```
#recuadro1{
  -ms-transform: scale(0.8 , 1.2);
  -webkit-transform: scale(0.8 , 1.2);
  -moz-transform: scale(0.8 , 1.2);
  -o-transform: scale(0.8 , 1.2);
  transform: scale(0.8 , 1.2);

  -ms-transform-origin: left top;
  -webkit-transform-origin: left top;
  -moz-transform-origin: left top;
  -o-transform-origin: left top;
  transform-origin: left top;
}
```

### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

```
border-radius: 20px;
background-color:#ddd;
width:200px;
padding:10px;
}
```

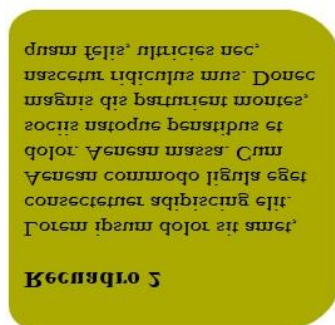
Si tenemos que escalar solo en x o en y podemos utilizar alguna de las dos funciones scaleX o scaleY.

```
Elemento {
    transform: scaleX(2);
}
Elemento {
    transform: scaleY(0.5);
}
```

Otra posibilidad es utilizar un valor negativo, lo que nos permite tener una reflexión del elemento. Por ejemplo un recuadro con reflexión en y:

```
#recuadro2{
    -ms-transform: scale(1.20 ,-1);
    -webkit-transform: scale(1.20 ,-1);
    -moz-transform: scale(1.20 ,-1);
    -o-transform: scale(1.20 ,-1);
    transform: scale(1.20 ,-1);

    border-radius: 20px 40px 40px 20px;
    background-color:#aa0;
    width:200px;
    padding:10px;
    margin-top:70px;
}
```



### 3.4. Translación (transform:translate)

La función translate permite desplazar un elemento HTML indicando una medida (en porcentaje, píxeles, em, etc.) la sintaxis es:

```
Elemento {
    transform: translate(valorx ,valory);
}
```

Por ejemplo si queremos trasladar 25 píxeles en "x" y 10 píxeles en "y" luego debemos codificar:

```
#recuadro1{
    -ms-transform: translate(25px,10px);
    -webkit-transform: translate(25px,10px);
    -moz-transform: translate(25px,10px);
    -o-transform: translate(25px,10px);
    transform: translate(25px,10px);
}
```

```

border-radius: 20px;
background-color:#ddd;
width:200px;
padding:10px;
}

#recuadro2{
border-radius: 20px;
background-color:#aa0;
width:200px;
padding:10px;
margin-top:50px;
}

```

Si vemos el primer recuadro tiene aplicada la traslación.

También disponemos de las funciones `translateX` y `translateY` para los casos donde solo debemos trasladar en x o y:

```

Elemento {
    transform: translateX(valor);
}
Elemento {
    transform: translateY(valor);
}

```

### 3.5. Torcer (`transform:skew`)

La función `skew` permite torcer el elemento HTML en X e Y, la sintaxis es la siguiente:

```

Elemento {
    transform: skew(gradosx ,gradosy);
}

```

Por ejemplo si queremos torcer en X 15 grados un recuadro luego debemos codificar:

```

#recuadro1{
-ms-transform: skew(15deg,0deg);
-webkit-transform: skew(15deg,0deg);
-moz-transform: skew(15deg,0deg);
-o-transform: skew(15deg,0deg);
transform: skew(15deg,0deg);
border-radius: 20px;
background-color:#ddd;
width:200px;
padding:10px;
}

```

Si queremos torcer en Y 15 grados un recuadro luego debemos codificar:

```
#recuadro2{
```

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

#### Recuadro 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

```

-ms-transform: skew(0deg,15deg);
-webkit-transform: skew(0deg,15deg);
-moz-transform: skew(0deg,15deg);
-o-transform: skew(0deg,15deg);
transform: skew(0deg,15deg);
border-radius: 20px;
background-color:#ddd;
width:200px;
padding:10px;
margin-top:50px;
}

```

#### Recuadro 2

Lorem ipsum dolor sit amet,  
 consectetur adipiscing elit.  
 Aenean commodo ligula eget  
 dolor. Aenean massa. Cum  
 sociis natoque penatibus et  
 magnis dis parturient montes,  
 nascetur ridiculus mus. Donec  
 quam felis, ultricies nec,

Por último si aplicamos tanto en X e Y tenemos:

```

#recuadro3{
  -ms-transform: skew(15deg,15deg);
  -webkit-transform: skew(15deg,15deg);
  -moz-transform: skew(15deg,15deg);
  -o-transform: skew(15deg,15deg);
  transform: skew(15deg,15deg);
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
  margin-top:70px;
}

```

#### Recuadro 3

Lorem ipsum dolor sit amet,  
 consectetur adipiscing elit.  
 Aenean commodo ligula eget  
 dolor. Aenean massa. Cum  
 sociis natoque penatibus et  
 magnis dis parturient montes,  
 nascetur ridiculus mus. Donec  
 quam felis, ultricies nec,

También disponemos de las funciones skewX y skewY para los casos donde solo debemos torcer en X o Y:

```

Elemento {
  transform: skewX(grados);
}
Elemento {
  transform: skewY(grados);
}

```

Los grados también pueden ser un valor negativo.

### 3.6. Múltiples transformaciones en forma simultánea

La sintaxis de CSS3 nos permite definir en la propiedad transform múltiples transformaciones en forma simultánea (debemos dejar al menos un espacio en blanco entre cada llamada a una función de transformación):

```

Elemento {
  transform: rotate(grados) scale(valorx ,valory)
  translate(valorx ,valory) skew(gradosx ,gradosy);
}

```

Por ejemplo apliquemos múltiples transformaciones a un recuadro:

```
#recuadro1{
  -moz-transform: rotate(15deg) scale(0.9,0.6) translate(10px,10px)
  skew(5deg,0deg);
  -ms-transform: rotate(15deg) scale(0.9,0.6) translate(10px,10px)
  skew(5deg,0deg);
  -webkit-transform: rotate(15deg) scale(0.9,0.6)
  translate(10px,10px) skew(5deg,0deg);
  -o-transform: rotate(15deg) scale(0.9,0.6) translate(10px,10px)
  skew(5deg,0deg);
  transform: rotate(15deg) scale(0.9,0.6) translate(10px,10px)
  skew(5deg,0deg);
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
}
```



## 4. Opacidad

---

### 4.1. Opacidad (opacity)

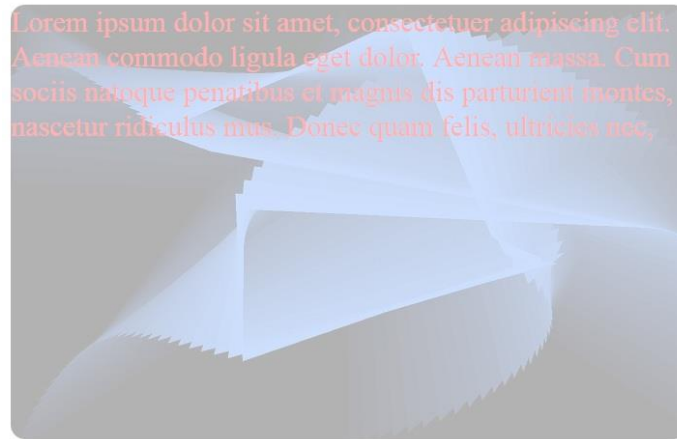
La opacidad es una característica de los objetos de no dejar pasar la luz (mientras un objeto es más opaco significa que no deja pasar la luz) Un elemento HTML dispone de la propiedad `opacity` para definir cual es su opacidad. La sintaxis es la siguiente:

```
Elemento {
  opacity: valor;
}
```

El valor es un número comprendido entre 0 y 1. El 0 significa que es totalmente transparente (luego no se verá nada en pantalla, pero el espacio ocupado por el elemento HTML queda reservado), el 1 significa que es totalmente opaco (no deja pasar la luz)

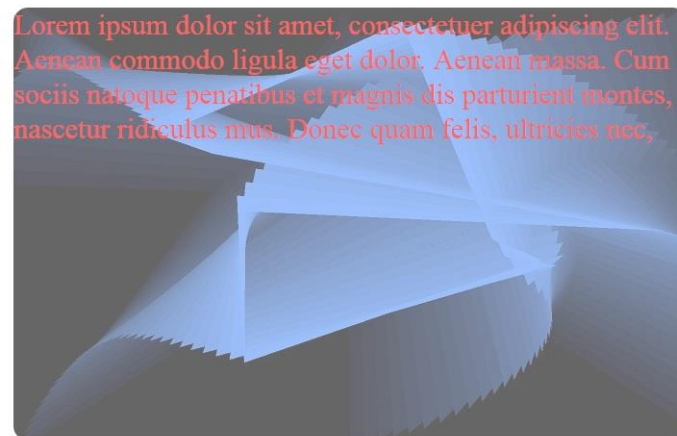
Veamos tres recuadros con una imagen de fondo y un texto en su interior con diferentes niveles de opacidad (tengamos en cuenta que cuando le asignamos una opacidad a un elemento HTML luego todos los elementos contenidos en dicho elementos heredan dicha opacidad):

```
#recuadro1 {
  background-image: url("foto1.jpg");
  opacity:0.3;
  color:#f00;
  width:700px;
  height:450px;
  border-radius:15px;
  font-size:30px;
}
```



Ahora con una opacidad mayor:

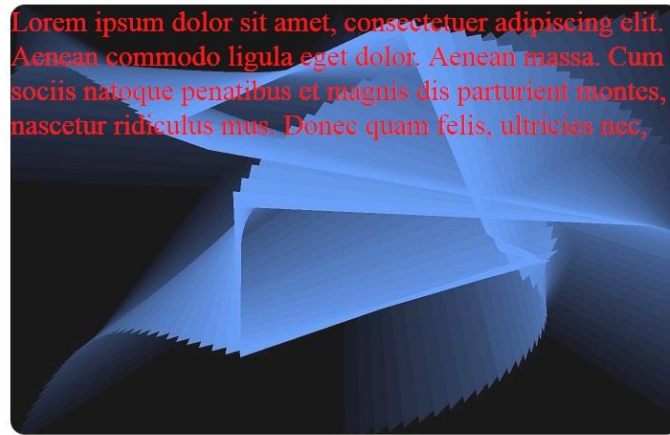
```
#recuadro2 {  
  background-image: url("foto1.jpg");  
  opacity:0.6;  
  color:#f00;  
  width:700px;  
  height:450px;  
  border-radius:15px;  
  font-size:30px;  
}
```



Finalmente, con una opacidad de 0.9 (casi no deja pasar nada de luz):

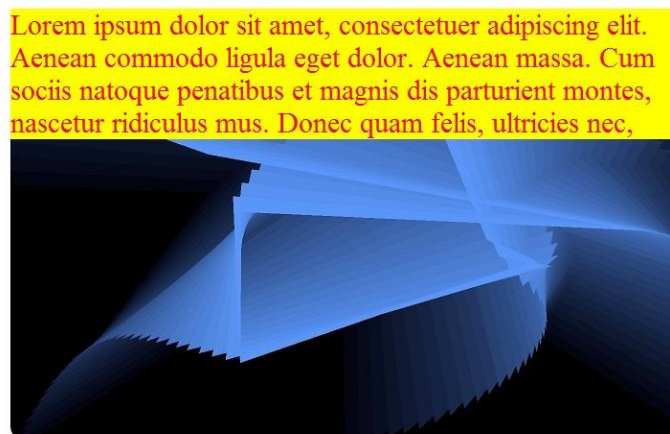
```
#recuadro3 {  
  background-image: url("foto1.jpg");  
  opacity:0.9;  
  color:#f00;  
  width:700px;  
  height:450px;  
  border-radius:15px;  
  font-size:30px;  
}
```





Veamos que pasa si no disponemos opacidad en el recuadro y el texto tiene un color de fondo:

```
#recuadro4 {  
  background-image: url("foto1.jpg");  
  color:#f00;  
  width:700px;  
  height:450px;  
  border-radius:15px;  
  font-size:30px;  
}  
.texto4 {  
  background-color:yellow;  
}
```



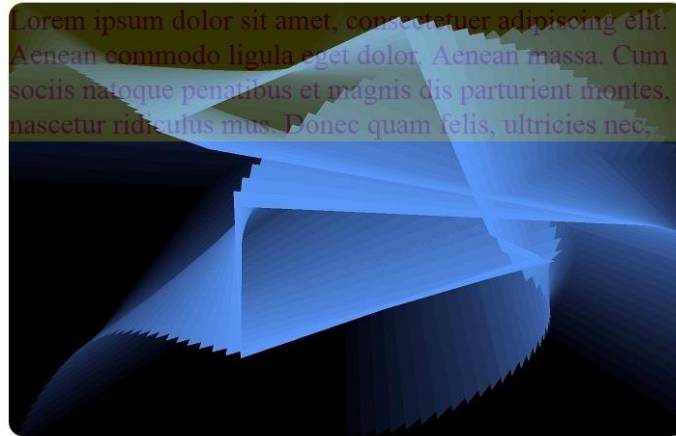
Y luego definiendo una opacidad en el párrafo:

```
#recuadro5 {  
  background-image: url("foto1.jpg");  
  color:#f00;  
  width:700px;  
  height:450px;  
  border-radius:15px;  
  font-size:30px;
```

```

}
.texto5 {
  background-color:yellow;
  opacity:0.2;
}

```



## 4.2. Opacidad (color)

En CSS3 aparece una variante de la asignación del color mediante la función rgba.

```

Elemento {
  color: rgba(rojo,verde,azul,opacidad);
}

```

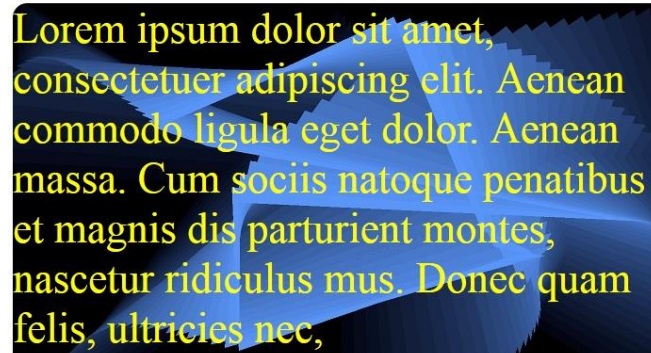
Para definir un color debemos indicar cuatro valores, los tres primeros son valores enteros entre 0 y 255, que indican la cantidad de rojo, verde y azul. El cuarto valor es un número entre 0 y 1 que indica la opacidad que se aplica al color. Si indicamos un 1 se grafica totalmente opaco (es el valor por defecto) un valor menor hará más transparente el gráfico.

Veamos un ejemplo primero dispondremos un texto dentro de un recuadro que tiene una imagen y no utilizaremos la función rgba para definir la opacidad, sino utilizaremos la función rgb para definir solo el color:

```

#recuadro1 {
  background-image: url("foto1.jpg");
  color:#f00;
  width:700px;
  height:450px;
  border-radius:15px;
  font-size:45px;
}
.texto1 {
  color:rgb(255,255,0);
}

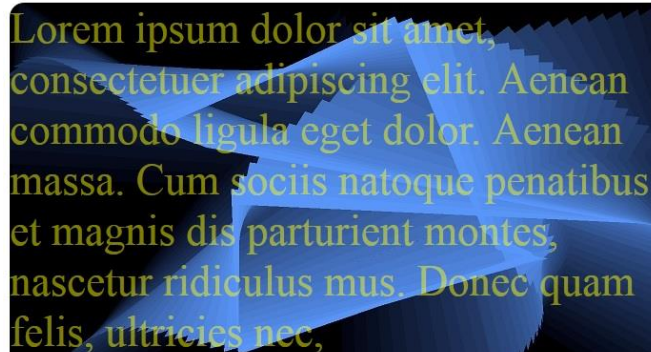
```



Lorem ipsum dolor sit amet,  
consectetur adipiscing elit. Aenean  
commodo ligula eget dolor. Aenean  
massa. Cum sociis natoque penatibus  
et magnis dis parturient montes,  
nascetur ridiculus mus. Donec quam  
felis, ultricies nec,

Y ahora si utilizaremos la función nueva rgba:

```
#recuadro2 {  
  background-image: url("foto1.jpg");  
  color:#f00;  
  width:700px;  
  height:450px;  
  border-radius:15px;  
  font-size:45px;  
  margin-top:10px;  
}  
.texto2 {  
  color:rgba(255,255,0,0.5);  
}
```



Lorem ipsum dolor sit amet,  
consectetur adipiscing elit. Aenean  
commodo ligula eget dolor. Aenean  
massa. Cum sociis natoque penatibus  
et magnis dis parturient montes,  
nascetur ridiculus mus. Donec quam  
felis, ultricies nec,

## 5. Columnas Múltiples

---

### 5.1. Columnas Múltiples (column-count)

Otra novedad que nos trae CSS3 es permitir disponer un bloque de texto en múltiples columnas con la única indicación de la cantidad de columnas que queremos que lo divida:

```
Elemento {
    column-count: cantidad de columnas;
}
```

Por ejemplo para generar un cuadro con tres columnas debemos implementar el siguiente código:

```
#recuadro1{
    -moz-column-count:3;
    -webkit-column-count:3;
    column-count:3;
    border-radius: 20px;
    background-color:#ddd;
    width:600px;
    padding:10px;
}
```

<b>Recuadro 1</b> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,	consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,	magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,
--	---	--

Tenemos que anteceder los prefijos -moz, -webkit para que funcionen con el Firefox y Chrome. El Opera ya implementa la propiedad definitiva column-count.

## 5.2. Múltiples columnas dinámicas (column-width)

Con la propiedad column-width también generamos múltiples columnas. Tiene sentido emplearla cuando nuestro diseño es flexible, es decir que si modificamos el tamaño de la ventana del navegador su contenido se relocaliza.

Con la propiedad column-width especificamos el ancho de la columna en píxeles, porcentaje etc. y luego se generan tantas columnas como entren en el contenedor.

La sintaxis es la siguiente:

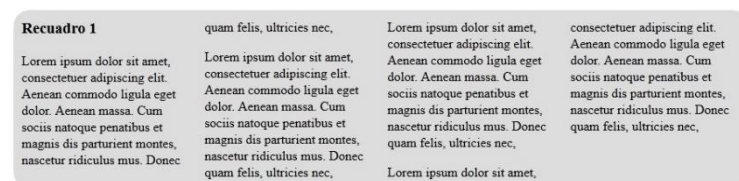
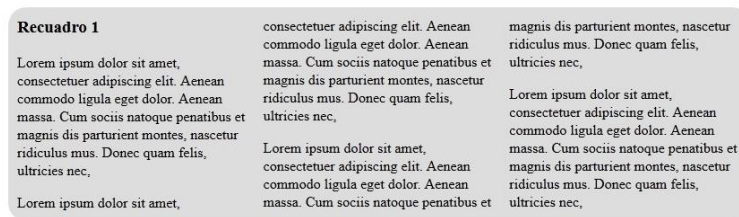
```
Elemento {
    column-width: ancho;
}
```

Si queremos crear un recuadro con texto y que cada columna ocupe 150 píxeles luego codificamos:

```
#recuadro1{
    -moz-column-width:200px;
    -webkit-column-width:200px;
    column-width:200px;
    border-radius: 20px;
    background-color:#ddd;
}
```

```
padding:10px;
}
```

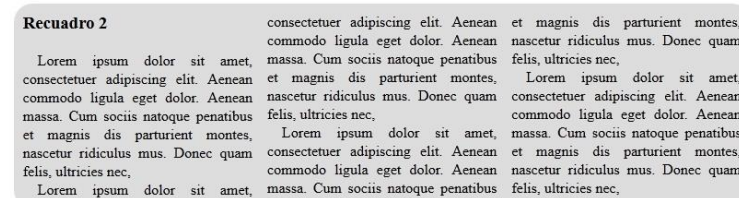
El resultado puede variar dependiendo las dimensiones de la ventana del navegador (siempre y cuando hayamos creado una página web fluida, es decir sin un ancho fijo):



Una mejora a la disposición del texto dentro de las columnas es inicializar la propiedad `text-indent` y `margin`:

```
#recuadro2 p {
  text-indent: 1em;
  margin: 0;
  text-align: justify;
}
```

Luego el resultado gráfico queda:



### 5.3. Múltiples columnas y su separación (column-gap)

El CSS3 nos permite especificar la separación entre las columnas. Su sintaxis es:

```
Elemento {
  column-gap: valor;
}
```

Podemos utilizar cualquier unidad para indicar la separación entre columnas (px, em etc.).

Si queremos una separación de 40 píxeles entre columnas luego debemos codificar:

```
#recuadro1{
  -moz-column-gap:40px;
  -webkit-column-gap:40px;
  column-gap:40px;
  -moz-column-count:3;
```



```

-webkit-column-count:3;
column-count:3;
border-radius: 20px;
background-color:#ddd;
width:600px;
padding:10px;
}

```

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

Lorem ipsum dolor sit amet, consectetur

adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et

magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

## 5.4. Múltiples columnas y línea separadora (column-rule)

Podemos configurar una línea separadora entre las columnas cuando utilizamos múltiples columnas. La sintaxis es la siguiente:

```

Elemento {
    column-rule: grosor estilo color;
}

```

Con el siguiente ejemplo veamos los distintos valores de esta propiedad. Definimos una línea de un píxel, con trazo sólido de color rojo:

```

#recuadro1{
    -moz-column-rule: 1px solid #f00;
    -webkit-column-rule: 1px solid #f00;
    column-rule: 1px solid #f00;
    -moz-column-count:3;
    -webkit-column-count:3;
    column-count:3;
    border-radius: 20px;
    background-color:#ddd;
    width:600px;
    padding:10px;
}

```

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

Lorem ipsum dolor sit amet,

consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

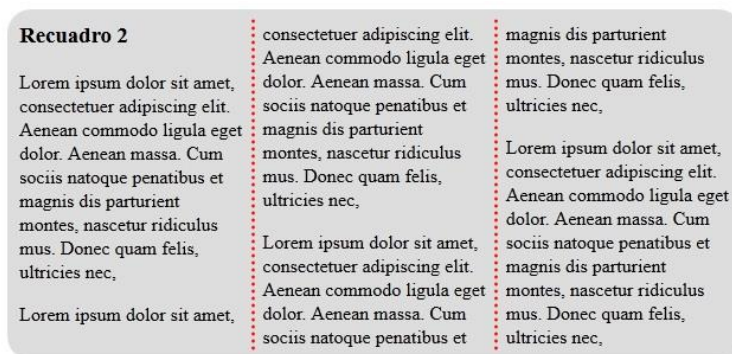
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et

magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

Otro ejemplo utilizando línea punteada:

```
#recuadro2{
  -moz-column-rule: 3px dotted #f00;
  -webkit-column-rule: 3px dotted #f00;
  column-rule: 3px dotted #f00;
  -moz-column-count:3;
  -webkit-column-count:3;
  column-count:3;
  border-radius: 20px;
  background-color:#ddd;
  width:600px;
  padding:10px;
  margin-top:10px;
}
```



Hay tres propiedades que permiten asignar en forma independiente el grosor, estilo y color del separador:

```
Elemento {
  column-rule-width: grosor;
  column-rule-style: estilo;
  column-rule-color: color;
}
```

Por ejemplo si queremos definir un separador con línea discontinua de 2 píxeles de color azul:

```
#recuadro3{
  -moz-column-rule-width: 2px;
  -webkit-column-rule-width: 2px;
  column-rule-width: 2px;
  -moz-column-rule-style: dashed;
  -webkit-column-rule-style: dashed;
  column-rule-style: dashed;
  -moz-column-rule-color: #00f;
  -webkit-column-rule-color: #00f;
  column-rule-color: #00f;
  -moz-column-count:3;
  -webkit-column-count:3;
  column-count:3;
  border-radius: 20px;
  background-color:#ddd;
}
```



```
width:600px;
padding:10px;
margin-top:10px;
}
```

<b>Recuadro 3</b>		
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et</p>	<p>consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et</p>	<p>magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,</p>

Si queremos una línea doble debemos indicar en el estilo el valor double:

```
#recuadro4{
  -moz-column-rule: 4px double #aaa;
  -moz-column-count:3;
  -webkit-column-count:3;
  column-count:3;
  border-radius: 20px;
  background-color:#ddd;
  width:600px;
  padding:10px;
  margin-top:10px;
}
```

<b>Recuadro 4</b>		
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et</p>	<p>consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et</p>	<p>magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,</p>

## 6. Importar una fuente no disponible en el navegador

En CSS3 se especifica una nueva sintaxis para importar fuentes no disponibles en el navegador y que se descargan de un servidor web.

El problema actual es que hay varios formatos de fuentes que son apoyados por distintos navegadores. El IE8 e inferiores solo admite el formato EOT que es propietario. El formato True Type es ampliamente soportado por navegadores modernos. Otros formatos son el OpenType, SVG y WOFF.

Esta diversidad de formatos de fuente hace necesario que indiquemos la mayor cantidad de formatos de fuentes posibles para nuestro sitio. Para importar una fuente que será utilizada por una página web tenemos que utilizar la regla @font-face, en la misma indicamos el nombre de la fuente a importar y la dirección web de donde la debe descargar el navegador, la sintaxis es:

```
@font-face {
  font-family: [nombre de la fuente];
  src: local(""),
  url("nombreakivo.woff") format("woff"),
  url("nombreakivo.otf") format("opentype"),
  url("nombreakivo.svg#nombre de la fuente") format("svg");
}
```

Podemos descargar una fuente, por ejemplo, del sitio <https://www.fontsquirrel.com/> para probar la importación de fuentes (hay una sección donde podemos descargar varios formatos para la misma fuente)

Veamos un ejemplo de utilizar una fuente descargada del sitio mencionado anteriormente:

```
<!DOCTYPE html>
<html>
<head>
<title>Prueba</title>
<style type="text/css">
@font-face {
  font-family: "Ubuntu";
  src: url("Ubuntu-Title-webfont.eot") format("eot"),
       url("Ubuntu-Title-webfont.woff") format("woff"),
       url("Ubuntu-Title-webfont.ttf") format("truetype"),
       url("Ubuntu-Title-webfont.svg#UbuntuTitle") format("svg");
}
#recuadro1{
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
}
#recuadro1 p {
  font-family:Ubuntu;
  color:#ff0000;
  font-size:20px;
}
body {
  background:white;
  margin:50px;
}
</style>
</head>
<body>
<div id="recuadro1">
<h3>Recuadro 1</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, </p>
</div>
</body>
</html>
```

El resultado tipográfico de importar esta fuente es:

En la regla `@font-face` hacemos referencia a cuatro fuentes con distintos formatos y definimos su nombre en la propiedad `font-family`:

```
@font-face {  
    font-family: "Ubuntu";  
    src: url("Ubuntu-Title-webfont.eot")  
    format("eot"),  
        url("Ubuntu-Title-webfont.woff")  
    format("woff"),  
        url("Ubuntu-Title-webfont.ttf") format("truetype"),  
        url("Ubuntu-Title-webfont.svg#UbuntuTitle") format("svg");  
}
```

Ahora podemos utilizar la fuente "Ubuntu" que acabamos de crear:

```
#recuadro1 p {  
    font-family:Ubuntu;  
    color:#ff0000;  
    font-size:20px;  
}
```

Con lo anterior indicamos que los párrafos contenidos en `#recuadro1` deben utilizar la fuente Ubuntu.

#### Recuadro 1

lorem ipsum dolor sit  
amet, consectetur  
adipiscing elit. aenean  
commodo ligula eget  
dolor. aenean massa.  
cum sociis natoque  
penatibus et magnis dis  
parturient montes,  
nascetur ridiculus mus.  
donec quam felis,  
ultrices nec.

## 7. Imágenes de fondo

---

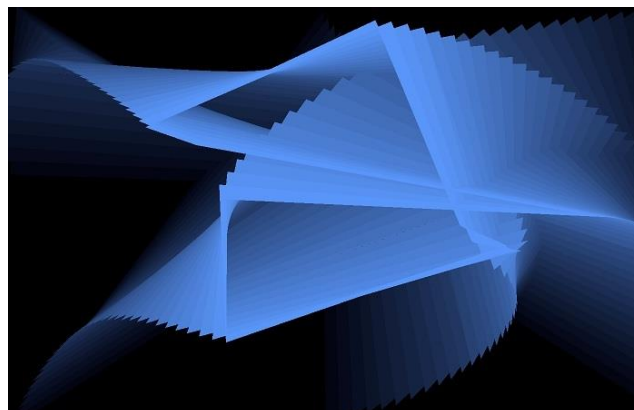
### 7.1. Imágenes de fondo (background-image)

La propiedad `background-image` permite insertar un conjunto de imágenes dentro de un elemento. Para ello debemos especificar sus nombres:

```
Elemento {  
    background-image: url("imagen1", url("imagen2", ...));  
}
```

La primera imagen que se dibuja es la que indicamos al final de la lista.

Por ejemplo vamos a mostrar una imagen de fondo con formato jpg y sobre esta una de tipo png, la primera:



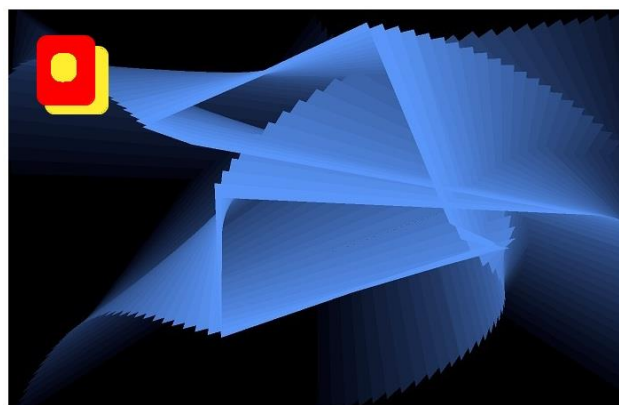
y la imagen png es:



Luego la página que muestra, superpuestas las dos imágenes en un recuadro es:

```
<!DOCTYPE html>
<html>
<head>
<title>Prueba</title>
<style type="text/css">
#recuadro1{
    background-image: url("logo1.png"), url("foto1.jpg");
    background-repeat: no-repeat;
    width:700px;
    height:450px;
}
body {
    background:white;
    margin:50px;
}
</style>
</head>
<body>
<div id="recuadro1">
</div>
</body>
</html>
```

El resultado es:

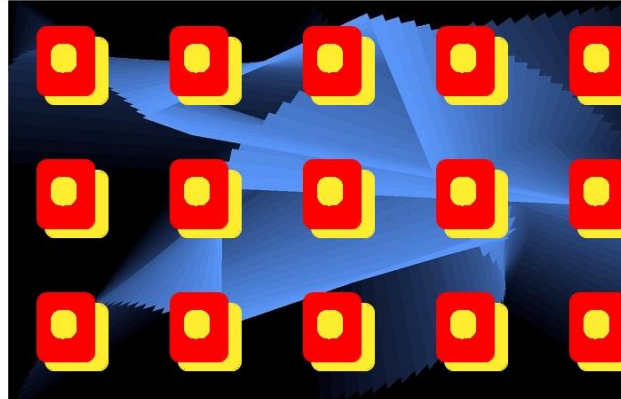


Para esto definimos:

```
#recuadro1{
    background-image: url("logo1.png"), url("foto1.jpg");
    background-repeat: no-repeat;
    width:700px;
    height:450px;
}
```

El recuadro coincide con el tamaño de la imagen "foto1.jpg" (700\*450) Luego la imagen "logo1.png" es de 150\*150 píxeles.

Como vemos primero se dibuja la imagen "foto1.jpg" (que es la última) y luego sobre esta la imagen "logo1.png". Otra cosa importante es inicializar la propiedad background-repeat con el valor no-repeat. En caso de no inicializar dicha propiedad tendremos a la imagen logo1.png repetida dentro del recuadro:



## 7.2. Imágenes de fondo en diferentes posiciones (background-position)

Podemos, mediante la propiedad background-position, indicar la posición de la imagen con respecto al contenedor.

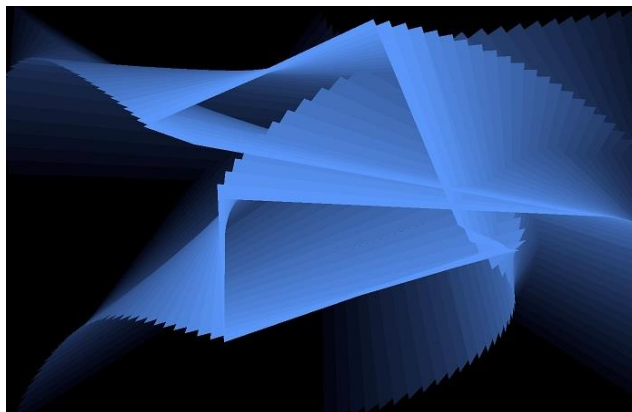
La sintaxis es la siguiente:

```
Elemento {  
    background-image:    url("imagen1", url("imagen2", ...));  
    background-position: posx posy , posx posy;  
}
```

El par de valores que indicamos coincide con la posición de las imágenes que especificamos en la propiedad background-image.

Implementaremos una página que muestre una imagen de 700\*400 y dentro de esta otra imagen que se ubique en los cuatro vértices de la imagen más grande.

Utilizaremos las imágenes:



y la imagen: 

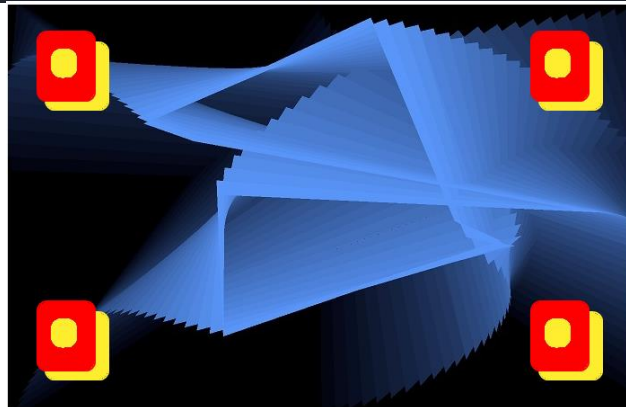
La página es:

```
<!DOCTYPE html>
<html>
<head>
<title>Prueba</title>
<style type="text/css">

#recuadro1{
    background-image:    url("logo1.png"),
                        url("logo1.png"),
                        url("logo1.png"),
                        url("logo1.png"),
                        url("foto1.jpg");

    background-position: 0% 0%,
                        100% 0%,
                        0% 100%,
                        100% 100%;

    background-repeat: no-repeat;
    width:700px;
    height:450px;
}
body {
    background:white;
    margin:50px;
}
</style>
</head>
<body>
<div id="recuadro1">
</div>
</body>
</html>
```



Hay que recordar que la primera imagen que se muestra es la que especificamos última en la propiedad background-image. Luego en la propiedad background-position indicamos en el mismo orden la ubicación de cada imagen. La primera la ubicamos en el vértice superior izquierdo ya que especificamos los valores 0% 0%. La segunda imagen la especificamos en el vértice superior derecho con los valores 100% 0%. Así indicamos las otras dos imágenes. La quinta imagen no es necesario indicar el background-position ya que el ancho y alto coincide con los valores asignados a las propiedades width y height.

```
#recuadro1{
    background-image:    url("logo1.png"),
                        url("logo1.png"),
```

```

        url("logo1.png"),
        url("logo1.png"),
        url("foto1.jpg");
background-position: 0% 0%,
                    100% 0%,
                    0% 100%,
                    100% 100%;
background-repeat: no-repeat;
width:700px;
height:450px;
}

```

### 7.3. Imágenes de fondo (background-size)

Esta propiedad nos permite escalar una imagen dispuesta en el fondo. La sintaxis es:

```

Elemento {
    background-size: ancho alto;
}

```

Estas longitudes se las puede especificar en píxeles, porcentajes etc.

Por ejemplo si queremos mostrar tres imágenes dentro de un div con distintos tamaño:

```

<!DOCTYPE html>
<html>
<head>
<title>Prueba</title>
<style type="text/css">
#recuadro1{
    background-image:    url("logo1.png"),
                        url("logo1.png"),
                        url("foto1.jpg");

    background-position: 0% 0%,
                        50% 50%;

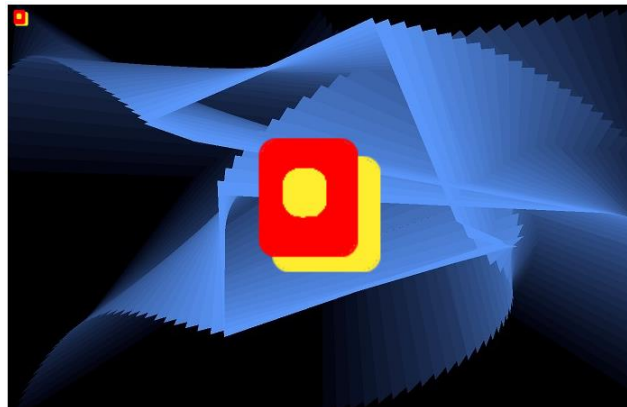
    background-size:     30px 30px,
                        250px 250px,
                        700px 450px;

    background-repeat: no-repeat;
    width:700px;
    height:450px;
}
body {
    background:white;
    margin:50px;
}
</style>
</head>
<body>
<div id="recuadro1">
</div>
</body>
</html>

```



Podemos ver que dibujamos tres imágenes "foto1.jpg" aparece en el fondo y luego dos veces mostramos "logo1.png" y le damos como tamaño al primero 30\*30 píxeles y al segundo 250\*250.



```
#recuadro1{
  background-image:    url("logo1.png"),
                      url("logo1.png"),
                      url("foto1.jpg");

  background-position: 0% 0%,
                      50% 50%;

  background-size:     30px 30px,
                      250px 250px,
                      700px 450px;

  background-repeat:   no-repeat;
  width:700px;
  height:450px;
}
```

Si queremos que una imagen tome el tamaño por defecto que tiene la imagen sin reescalar podemos utilizar la palabra clave "auto", por ejemplo para mostrar la primera imagen con el tamaño original podemos escribir:

```
#recuadro1{
  background-image:    url("logo1.png"),
                      url("logo1.png"),
                      url("foto1.jpg");

  background-position: 0% 0%,
                      50% 50%;

  background-size:     30px 30px,
                      250px 250px,
                      auto auto;

  background-repeat:   no-repeat;
  width:700px;
  height:450px;
}
```

Si utilizamos como unidad porcentajes, la dimensión se basa en el elemento contenedor.

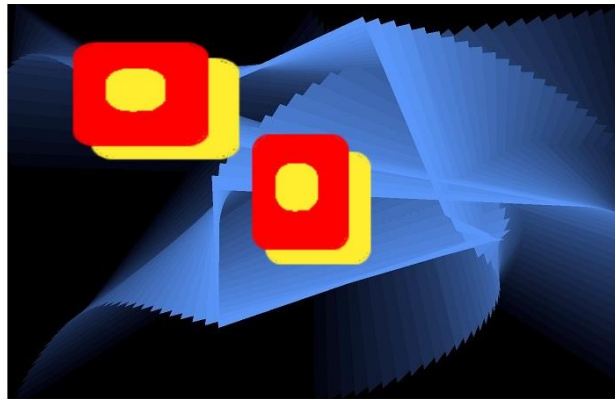
Así, por ejemplo, un ancho y un alto de 50% hará que el fondo de la imagen llenará el recipiente 1/4:

```
#recuadro1{
  background-image:    url("logo1.png"),
                      url("logo1.png"),
                      url("foto1.jpg");

  background-position: 0% 0%,
                      50% 50%;

  background-size:     50% 50%,
                      250px 250px,
                      auto auto;

  background-repeat: no-repeat;
  width:700px;
  height:450px;
}
```



#### 7.4. Imágenes de fondo (background-origin)

La propiedad `background-origin` establece el punto donde se comienza a dibujar el fondo. Pudiendo ser en el borde, la almohadilla o el contenido (`border-box`, `padding-box` o `content-box`)

```
Elemento {
  background-origin: border-box/padding-box/content-box;
}
```

Un ejemplo de disponer en tres recuadros imágenes de fondo inicializando la propiedad `background-origin` con los tres valores posibles:

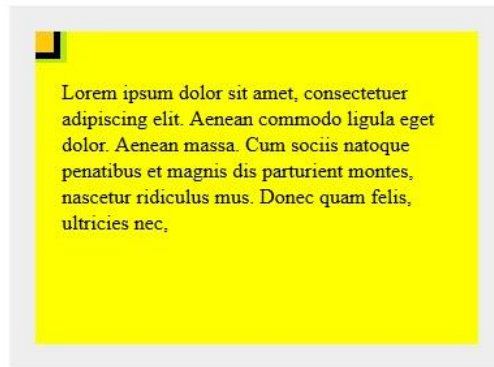
```
<!DOCTYPE html>
<html>
<head>
<title>Prueba</title>
<style type="text/css">
#recuadro1{
  background-origin: border-box;
  border: 20px solid #eee;
  padding: 20px;
  background-image: url("casa.png");
  background-color:#ff0;
  background-repeat: no-repeat;
  width:300px;
  height:200px;
}
```

```

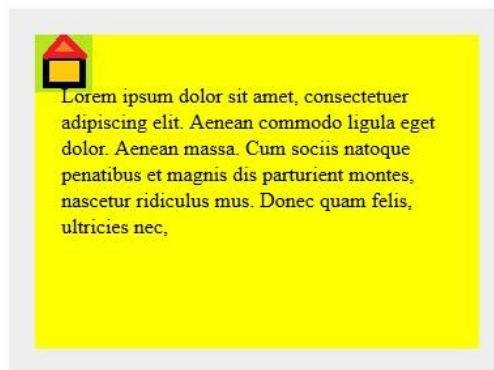
#recuadro2{
  background-origin: padding-box;
  border: 20px solid #eee;
  padding: 20px;
  background-image: url("casa.png");
  background-color:#ff0;
  background-repeat: no-repeat;
  width:300px;
  height:200px;
  margin-top:50px;
}
#recuadro3{
  background-origin: content-box;
  border: 20px solid #eee;
  padding: 20px;
  background-image: url("casa.png");
  background-color:#ff0;
  background-repeat: no-repeat;
  width:300px;
  height:200px;
  margin-top:50px;
}
body {
  background:white;
  margin:50px;
}
</style>
</head>
<body>
<div id="recuadro1">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo
ligula eget dolor.
Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes,
nascetur ridiculus mus.
  Donec quam felis, ultricies nec, </p>
</div>
<div id="recuadro2">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo
ligula eget dolor.
  Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes,
nascetur ridiculus mus.
  Donec quam felis, ultricies nec, </p>
</div>
<div id="recuadro3">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo
ligula eget dolor.
  Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes,
nascetur ridiculus mus.
  Donec quam felis, ultricies nec, </p>
</div>
</body>
</html>

```

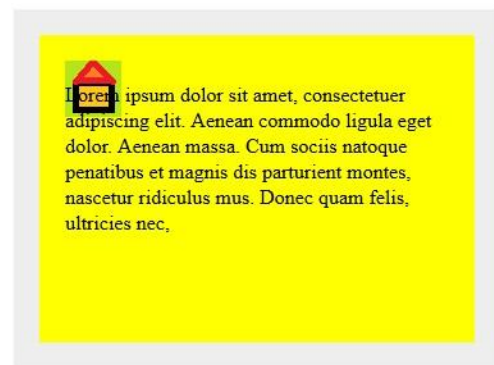
Con background-origin: border-box; el resultado es:



Con background-origin: padding-box; el resultado es:



Con background-origin: content-box; el resultado es:



## 8. Transiciones

---

### 8.1. Transiciones de una propiedad (transition)

Las transiciones en CSS3 permiten modificar el valor de una propiedad de un elemento HTML en forma gradual durante un tiempo determinado de un estado inicial a un estado final.

La sintaxis más simple para definir una transición de una propiedad es:

```
Elemento {  
    transition: [nombre de propiedad] [duración de la transición];  
}
```

Por ejemplo crearemos dos recuadros y cuando pasemos la flecha del mouse sobre el mismo cambiaremos su tamaño. El primer recuadro lo haremos sin transición:

```
#recuadro1{
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  padding:10px;
}
#recuadro1:hover{
  width:300px;
}
```

Cuando probamos la página veremos que el recuadro1 cambia el ancho del recuadro de 200 píxeles a 300 píxeles en forma instantánea al pasar la flecha del mouse.

Ahora si definimos la propiedad transition indicando que actúe sobre la propiedad width y que el cambio lo realice en 1 segundo:

```
#recuadro2{
  border-radius: 20px;
  background-color:#ff0;
  width:200px;
  padding:10px;
  transition:width 1s;
  -moz-transition:width 1s;
  -ms-transition:width 1s;
  -webkit-transition:width 1s;
  -o-transition:width 1s;
}
#recuadro2:hover{
  width:300px;
}
```

Luego tenemos la siguiente sintaxis:

`transition:width 1s;`

Como vemos indicamos que la transición afecta a la propiedad width y que el cambio se realice en 1 segundo. Debemos agregar la propiedad con cada prefijo de navegador hasta que se estabilice su uso.

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

#### Recuadro 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

#### Recuadro 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

#### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

#### Recuadro 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

## 8.2. Transiciones de múltiples propiedades (transition)

También podemos hacer transiciones de múltiples propiedades, para ello indicamos cada transición separada por coma:

```
Elemento {  
    transition: [nombre de propiedad] [duración de la transición],  
               [nombre de propiedad] [duración de la transición],  
               [nombre de propiedad] [duración de la transición];  
}
```

Por ejemplo si queremos que el recuadro modifique su ancho y cambie de color luego debemos codificar lo siguiente:

```
#recuadro1{  
    border-radius: 20px;  
    background-color:#ff0;  
    width:200px;  
    padding:10px;  
    transition:width 1s,  
               background-color 8s;  
    -moz-transition:width 1s,  
                   background-color 8s;  
    -ms-transition:width 1s,  
                   background-color 8s;  
    -webkit-transition:width 1s,  
                       background-color 8s;  
    -o-transition:width 1s,  
                  background-color 8s;  
}  
#recuadro1:hover{  
    width:300px;  
    background-color: #f00;  
}
```

Como vemos en la propiedad transition indicamos la propiedad width con una duración de un segundo y la propiedad background-color con un valor de 8 segundos. Esto significa que cuando dispongamos la flecha del mouse dentro del div se lanzarán ambas transiciones que tienen duraciones distintas (cambiará de 200 píxeles a 300 píxeles en el lapso de un segundo y también cambiará del color amarillo al rojo en forma gradual en un lapso de 8 segundos).

### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

### Recuadro 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,

## 8.3. Transiciones (funciones de transición)

Un tercer parámetro opcional de la propiedad transition es indicar una "función de transición" que nos permite seleccionar la velocidad durante la transición:

```
Elemento {  
    transition: [nombre de propiedad] [duración de la transición]  
    [función de transición];  
}
```

Los valores posibles que podemos especificar son:

- **ease**: Define un efecto de transición con un comienzo lento, luego rápido y finalmente termina lento (cuando no definimos la función de transición elige esta por defecto)
- **linear**: Define un efecto de transición con la misma velocidad de inicio a fin.
- **ease-in**: Define un efecto de transición con un comienzo lento.
- **ease-out**: Define un efecto de transición con un final lento.
- **ease-in-out**: Define un efecto de transición con un comienzo lento y un final lento.

Un ejemplo en el que se incluye la sintaxis de todos estos valores que se ha definido podría ser:

```
<!DOCTYPE html>  
<html>  
<head>  
<title>Prueba</title>  
<style type="text/css">  
#recuadro1{  
    margin: 5px;  
    border-radius: 20px;  
    background-color:#eee;  
    width:200px;  
    padding:10px;  
    transition:width 2s ease;  
    -moz-transition:width 2s ease;  
    -ms-transition:width 2s ease;  
    -webkit-transition:width 2s ease;  
    -o-transition:width 2s ease;  
}  
#recuadro1:hover{  
    width:300px;  
}  
#recuadro2{  
    margin: 5px;  
    border-radius: 20px;  
    background-color:#eee;  
    width:200px;  
    padding:10px;  
    transition:width 2s linear;  
    -moz-transition:width 2s linear;  
    -ms-transition:width 2s linear;  
    -webkit-transition:width 2s linear;  
    -o-transition:width 2s linear;  
}  
#recuadro2:hover{  
    width:300px;  
}  
#recuadro3{  
    margin: 5px;  
    border-radius: 20px;
```



```

background-color:#eee;
width:200px;
padding:10px;
transition:width 2s ease-in;
-moz-transition:width 2s ease-in;
-ms-transition:width 2s ease-in;
-webkit-transition:width 2s ease-in;
-o-transition:width 2s ease-in;
}
#recuadro3:hover{
width:300px;
}
#recuadro4{
margin: 5px;
border-radius: 20px;
background-color:#eee;
width:200px;
padding:10px;
transition:width 2s ease-out;
-moz-transition:width 2s ease-out;
-ms-transition:width 2s ease-out;
-webkit-transition:width 2s ease-out;
-o-transition:width 2s ease-out;
}
#recuadro4:hover{
width:300px;
}
#recuadro5{
margin: 5px;
border-radius: 20px;
background-color:#eee;
width:200px;
padding:10px;
transition:width 2s ease-in-out;
-moz-transition:width 2s ease-in-out;
-ms-transition:width 2s ease-in-out;
-webkit-transition:width 2s ease-in-out;
-o-transition:width 2s ease-in-out;
}
#recuadro5:hover{
width:300px;
}

body {
background:white;
margin:50px;
}
</style>
</head>
<body>
<div id="recuadro1">
<h3>ease</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,
</p>
</div>
<div id="recuadro2">
<h3>linear</h3>

```

```

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo
ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis
parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,
</p>
</div>
<div id="recuadro3">
<h3>ease-in</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo
ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis
parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,
</p>
</div>
<div id="recuadro4">
<h3>ease-out</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo
ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis
parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,
</p>
</div>
<div id="recuadro5">
<h3>ease-in-out</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo
ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis
parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec,
</p>
</div>
</body>
</html>

```

#### 8.4. Transiciones (tiempo de demora en iniciar la transición)

El cuarto parámetro opcional de la propiedad `transition` es indicar un tiempo de espera hasta que se inicie la transición:

```

Elemento {
    transition: [nombre de propiedad] [duración de la transición]
               [función de transición] [tiempo de inicio];
}

```

Es decir, indicamos la cantidad de milisegundos o segundos hasta que se inicia el proceso de transición.

Por ejemplo si queremos que la transición comience en 1 segundo la sintaxis será la siguiente:

```

#recuadro1{
    border-radius: 20px;
    background-color:#eee;
    width:200px;
    padding:10px;
    transition:width 1s ease 1s;
    -moz-transition:width 1s ease 1s;
    -ms-transition:width 1s ease 1s;
    -webkit-transition:width 1s ease 1s;
    -o-transition:width 1s ease 1s;
}

```

Puede ser muy útil si queremos encadenar transiciones, por ejemplo si queremos que el elemento modifique su ancho y luego al finalizar el cambio del ancho modifique su color:

```
#recuadro2{
  margin:5px;
  border-radius: 20px;
  background-color:#eee;
  width:200px;
  padding:10px;
  transition:width 1s ease,
              background-color 1s ease 1s;
  -moz-transition:width 1s ease,
                  background-color 1s ease 1s;
  -ms-transition:width 1s ease,
                  background-color 1s ease 1s;
  -webkit-transition:width 1s ease,
                     background-color 1s ease 1s;
  -o-transition:width 1s ease,
                 background-color 1s ease 1s;
}

#recuadro2:hover{
  width:300px;
  background-color:#ff0;
}
```

## 8.5. Transiciones (otra sintaxis para hacer lo mismo)

Hemos visto que la propiedad `transition` podemos indicarle hasta cuatro parámetros. Los dos primeros son obligatorios y los otros son opcionales.

CSS3 dispone de otras cuatro funciones para indicar cada uno de estos valores en forma independiente:

```
Elemento {
  transition-property: [nombre de propiedad];
  transition-duration: [duración de la transición];
  transition-timing-function: [función de transición];
  transition-delay: [tiempo de inicio];
}
```

Por ejemplo si queremos modificar el ancho de un elemento cuando disponemos la flecha del mouse y que dure 1 segundo y se lance luego de medio segundo la sintaxis es la siguiente:

```
#recuadro1{
  margin:5px;
  border-radius: 20px;
  background-color:#eee;
  width:200px;
  padding:10px;
  transition-property: width;
  transition-duration: 1s;
```

```

transition-timing-function: linear;
transition-delay: 0.5s;
-moz-transition-property: width;
-moz-transition-duration: 1s;
-moz-transition-timing-function: linear;
-moz-transition-delay: 0.5s;
-ms-transition-property: width;
-ms-transition-duration: 1s;
-ms-transition-timing-function: linear;
-ms-transition-delay: 0.5s;
-webkit-transition-property: width;
-webkit-transition-duration: 1s;
-webkit-transition-timing-function: linear;
-webkit-transition-delay: 0.5s;
-o-transition-property: width;
-o-transition-duration: 1s;
-o-transition-timing-function: linear;
-o-transition-delay: 0.5s;
}

#recuadro1:hover{
    width:300px;
}

```

Si queremos luego modificar más de una propiedad debemos indicarlos separando por coma:

```

#recuadro2{
    margin:5px;
    border-radius: 20px;
    background-color:#eee;
    width:200px;
    padding:10px;
    transition-property: width, background-color;
    transition-duration: 1s, 1s;
    transition-timing-function: linear, linear;
    transition-delay: 0s, 1s;
    -moz-transition-property: width, background-color;
    -moz-transition-duration: 1s, 1s;
    -moz-transition-timing-function: linear, linear;
    -moz-transition-delay: 0s, 1s;
    -ms-transition-property: width, background-color;
    -ms-transition-duration: 1s, 1s;
    -ms-transition-timing-function: linear, linear;
    -ms-transition-delay: 0s, 1s;
    -webkit-transition-property: width, background-color;
    -webkit-transition-duration: 1s, 1s;
    -webkit-transition-timing-function: linear, linear;
    -webkit-transition-delay: 0s, 1s;
    -o-transition-property: width, background-color;
    -o-transition-duration: 1s, 1s;
    -o-transition-timing-function: linear, linear;
}

```

```

    -o-transition-delay: 0s, 1s;
}

#recuadro2:hover{
    width:300px;
    background-color:#ff0;
}

```

Es importante el orden de las propiedades definidas en "transition-property", en este ejemplo definimos los valores width y background-color, luego por ejemplo en la propiedad "transition-delay" el valor de cero segundos se aplica a la propiedad width y el de 1 segundo a la propiedad "background-color".

## 9. Animaciones

---

### 9.1. Animaciones (sintaxis básica)

Las animaciones en CSS3 nos permiten hacer cosas que con las transiciones no alcanzamos o nos quedamos cortos.

La sintaxis básica para una animación:

```

Elemento {
    animation-name: [nombre de la animación];
    animation-duration: [tiempo de duración];
}
@ keyframes [nombre de la animación] {
    from {
        [propiedades y valores del estado inicial de la animación]
    }
    to {
        [propiedades y valores del estado final de la animación]
    }
}

```

Veamos un ejemplo donde emplearemos la sintaxis expuesta. Mostraremos un recuadro con un texto que parta de la columna 30px y avance hasta la columna 300px y que cambie del color gris al color amarillo:

```

#recuadro1{
    left:30px;
    position:relative;
    border-radius: 20px;
    background-color:#ddd;
    width:200px;
    height:100px;
    padding:4px;
    -moz-animation-name: animacion1;
    -moz-animation-duration: 4s;
    -webkit-animation-name: animacion1;
    -webkit-animation-duration: 4s;
    -o-animation-name: animacion1;
    -o-animation-duration: 4s;
}

```

```

}

@-moz-keyframes animacion1 {
  from {
    left:30px;
    background-color:#ddd;
  }
  to {
    left:300px;
    background-color:#ff0;
  }
}
@-webkit-keyframes animacion1 {
  from {
    left:30px;
    background-color:#ddd;
  }
  to {
    left:300px;
    background-color:#ff0;
  }
}
@-o-keyframes animacion1 {
  from {
    left:30px;
    background-color:#ddd;
  }
  to {
    left:300px;
    background-color:#ff0;
  }
}
}

```

**Título**

Lorem ipsum dolor sit amet,  
consectetuer adipiscing elit.

**Título**

Lorem ipsum dolor sit amet,  
consectetuer adipiscing elit.

**Título**

Lorem ipsum dolor sit amet,  
consectetuer adipiscing elit.

Como podemos ver inicializamos las propiedades animation-name con el nombre de la animación:

```

-moz-animation-name: animacion1;
-webkit-animation-name: animacion1;
-o-animation-name: animacion1;

```

Y la propiedad animation-duration con el tiempo duración de la animación:

```

-moz-animation-duration: 4s;
-webkit-animation-duration: 4s;
-o-animation-duration: 4s;

```

La sintaxis para especificar la animación propiamente dicha es idéntica para cada navegador pero como debemos utilizar el prefijo luego creamos tres:

```
@-moz-keyframes animacion1 {
  from {
    left:30px;
    background-color:#ddd;
  }
  to {
    left:300px;
    background-color:#ff0;
  }
}
```

En la parte del from indicamos el estado inicial del elemento (en este caso 30 píxeles respecto a la izquierda y de color gris (#ddd)), luego en la sección del “to” indicamos los valores finales que debe tomar el elemento (300 píxeles respecto a la izquierda y de color amarillo(#ff0))

## 9.2. Animaciones (animation-iteration-count y animation-direction)

Otras dos propiedades muy importantes para crear animaciones con CSS3 son animation-iteration-count y animation-direction:

```
Elemento {
  animation-iteration-count: [cantidad de veces a repetir la
animación o "infinite"];
  animation-direction: ["normal" o "alternate"];
}
```

Si queremos que la animación se repita solo tres veces, indicamos en la propiedad animation-iteration-count dicho valor:

```
-moz-animation-iteration-count: 3;
```

En cambio si queremos que se repita siempre, especificamos el valor "infinite":

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

La propiedad animation-direction indica como debe repetirse la animación, puede tomar el valor "normal" con lo que la animación cada vez que finaliza comienza desde el principio. Pero si inicializamos con el valor "alternate" cuando finaliza la animación comienza desde el final hasta el principio:

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

El ejemplo del concepto anterior que muestra un recuadro con un texto que se desplaza de izquierda a derecha cambiando su color, pero ahora indicando que se repita en forma indefinida y con el valor "alternate" en la propiedad animation-direction:

```
#recuadro1{
  left:30px;
  position:relative;
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  height:100px;
  padding:4px;
  -moz-animation-name: animacion1;
```



```

-moz-animation-duration: 4s;
-moz-animation-iteration-count: infinite;
-moz-animation-direction: alternate;
-webkit-animation-name: animacion1;
-webkit-animation-duration: 4s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-direction: alternate;
-o-animation-name: animacion1;
-o-animation-duration: 4s;
-o-animation-iteration-count: infinite;
-o-animation-direction: alternate;
}

@-moz-keyframes animacion1 {
  from {
    left:30px;
    background-color:#ddd;
  }
  to {
    left:300px;
    background-color:#ff0;
  }
}

@-webkit-keyframes animacion1 {
  from {
    left:30px;
    background-color:#ddd;
  }
  to {
    left:300px;
    background-color:#ff0;
  }
}

@-o-keyframes animacion1 {
  from {
    left:30px;
    background-color:#ddd;
  }
  to {
    left:300px;
    background-color:#ff0;
  }
}

```

### 9.3. Animaciones (animation-timing-function y animation-delay)

Similar a las transiciones disponemos de dos propiedades para definir la función de transición y el tiempo que debe esperar para comenzar la animación:

Elemento {

```

    animation-timing-function: [función de transición];
    animation-delay: [tiempo de demora para iniciar la animación];
}

```

Los valores posibles para la propiedad animation-timing-function:

- **ease**: Define un efecto de animación con un comienzo lento, luego rápido y finalmente termina lento (cuando no definimos la función elige esta por defecto)
- **linear**: Define un efecto de animación con la misma velocidad de inicio a fin.
- **ease-in**: Define un efecto de animación con un comienzo lento.
- **ease-out**: Define un efecto de animación con un final lento.
- **ease-in-out**: Define un efecto de animación con un comienzo lento y un final lento.

En el ejemplo del concepto anterior definimos la función de animación "linear" y un tiempo de retardo de 2 segundos:

```

#recuadro1{
  left:30px;
  position:relative;
  border-radius: 20px;
  background-color:#ddd;
  width:200px;
  height:100px;
  padding:4px;
  -moz-animation-name: animacion1;
  -moz-animation-duration: 4s;
  -moz-animation-iteration-count: infinite;
  -moz-animation-direction: alternate;
  -moz-animation-timing-function: linear;
  -moz-animation-delay: 2s;
  -webkit-animation-name: animacion1;
  -webkit-animation-duration: 4s;
  -webkit-animation-iteration-count: infinite;
  -webkit-animation-direction: alternate;
  -webkit-animation-timing-function: linear;
  -webkit-animation-delay: 2s;
  -o-animation-name: animacion1;
  -o-animation-duration: 4s;
  -o-animation-iteration-count: infinite;
  -o-animation-direction: alternate;
  -o-animation-timing-function: linear;
  -o-animation-delay: 2s;
}

@-moz-keyframes animacion1 {
  from {
    left:30px;
    background-color:#ddd;
  }
  to {

```

```

        left:300px;
        background-color:#ff0;
    }
}

@-webkit-keyframes animacion1 {
    from {
        left:30px;
        background-color:#ddd;
    }
    to {
        left:300px;
        background-color:#ff0;
    }
}

@-o-keyframes animacion1 {
    from {
        left:30px;
        background-color:#ddd;
    }
    to {
        left:300px;
        background-color:#ff0;
    }
}

```

#### 9.4. Animaciones (animation-play-state)

La propiedad `animation-play-state` nos permite cambiar el estado de la animación, los valores posibles son "running" y "paused", es decir ejecutándose o pausada. La sintaxis luego es:

```

Elemento {
    animation-play-state: [running o paused];
}

```

Por defecto una animación comienza con esta propiedad con el valor "running".

El siguiente ejemplo muestra un recuadro que cuando disponemos la flecha del mouse en su interior se activa la animación cambiando sus esquinas por esquinas redondeadas y el color interior:

```

#recuadro1{
    border-radius: 20px;
    background-color:#ddd;
    width:200px;
    height:100px;
    padding:8px;
    -moz-animation-play-state:paused;
    -moz-animation-name: animacion1;
    -moz-animation-duration: 4s;
    -moz-animation-iteration-count: infinite;
    -moz-animation-direction: alternate;
    -webkit-animation-play-state:paused;
}

```

```
-webkit-animation-name: animacion1;
-webkit-animation-duration: 4s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-direction: alternate;
-o-animation-play-state: paused;
-o-animation-name: animacion1;
-o-animation-duration: 4s;
-o-animation-iteration-count: infinite;
-o-animation-direction: alternate;
}

#recuadro1:hover {
  -moz-animation-play-state: running;
  -webkit-animation-play-state: running;
  -o-animation-play-state: running;
}

@-moz-keyframes animacion1 {
  from {
    border-radius: 0px;
    background-color: #ddd;
  }
  to {
    border-radius: 40px;
    background-color: #ff0;
  }
}

@-webkit-keyframes animacion1 {
  from {
    border-radius: 0px;
    background-color: #ddd;
  }
  to {
    border-radius: 40px;
    background-color: #ff0;
  }
}

@-o-keyframes animacion1 {
  from {
    border-radius: 0px;
    background-color: #ddd;
  }
  to {
    border-radius: 40px;
    background-color: #ff0;
  }
}
```

#### Título

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

#### Título

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

#### Título

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

#### Título

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

## 9.5. Animaciones (definición de más de 2 keyframes)

Hasta ahora hemos indicado en la animación solo 2 keyframes (el inicial y el final), pero para animaciones más complejas es posible que necesitemos más de 2 keyframes, para esto tenemos la siguiente sintaxis:

```
@-moz-keyframes [nombre de la animación] {
  0%{
    [propiedades y valores]
  }
  [valor en porcentaje]%{
    [propiedades y valores]
  }
  [valor en porcentaje]%{
    [propiedades y valores]
  }
  100%{
    [propiedades y valores]
  }
}
```

Por ejemplo si queremos 5 frames claves que nos permitan desplazar un recuadro sobre un perímetro de un rectángulo, tenemos que especificar los siguientes frames:

```
#recuadro1{
  left:30px;
  top:30px;
  position:relative;
  border-radius: 20px;
  background-color:#ddd;
  width:100px;
  height:100px;
  padding:4px;
  -moz-animation-name: animacion1;
  -moz-animation-duration: 4s;
  -moz-animation-iteration-count: infinite;
  -webkit-animation-name: animacion1;
  -webkit-animation-duration: 4s;
  -webkit-animation-iteration-count: infinite;
  -o-animation-name: animacion1;
  -o-animation-duration: 4s;
  -o-animation-iteration-count: infinite;
}

@-moz-keyframes animacion1 {
  0%{
```

```

        left:30px;
        top:30px;
    }
    25%{
        left:300px;
        top:30px;
    }
    50%{
        left:300px;
        top:200px;
    }
    75%{
        left:30px;
        top:200px;
    }
    100%{
        left:30px;
        top:30px;
    }
}

@-webkit-keyframes animacion1 {
    0%{
        left:30px;
        top:30px;
    }
    25%{
        left:300px;
        top:30px;
    }
    50%{
        left:300px;
        top:200px;
    }
    75%{
        left:30px;
        top:200px;
    }
    100%{
        left:30px;
        top:30px;
    }
}

@-o-keyframes animacion1 {
    0%{
        left:30px;
        top:30px;
    }
    25%{
        left:300px;

```

```

    top:30px;
  }
  50%{
    left:300px;
    top:200px;
  }
  75%{
    left:30px;
    top:200px;
  }
  100%{
    left:30px;
    top:30px;
  }
}

```

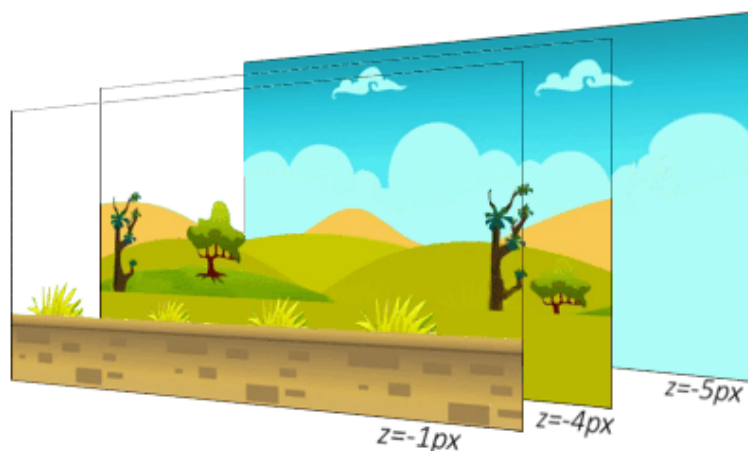
## 10. Parallax: perspectiva 3D

Seguramente has visto el efecto parallax en muchas páginas: cuando se ve que el fondo de la página está quieto, o se mueve muy lentamente, mientras el contenido de la página se desplaza al hacer scroll.

En la vida real este efecto es el que nos permite estimar distancias a ojo. Es lo que podríamos llamar también la perspectiva.

No voy a entrar en tecnicismos, solo decir que se trata de un efecto óptico que nos hace ver la imagen cercana más grande y con movimiento más rápido que los objetos lejanos cuando nos desplazamos. En cine y en animación es muy habitual verlo.

En cualquier elemento de la página web con fondo (background) tenemos dos capas superpuestas: el fondo y el contenido del elemento. Podríamos aplicar un efecto parallax de fondo, aunque realmente no definimos una perspectiva, pues ambas capas están prácticamente pegadas. No obstante, podemos aplicarle el principio del efecto parallax. Algo tan fácil como poner una posición fija para el fondo (**background-attachment: fixed**).

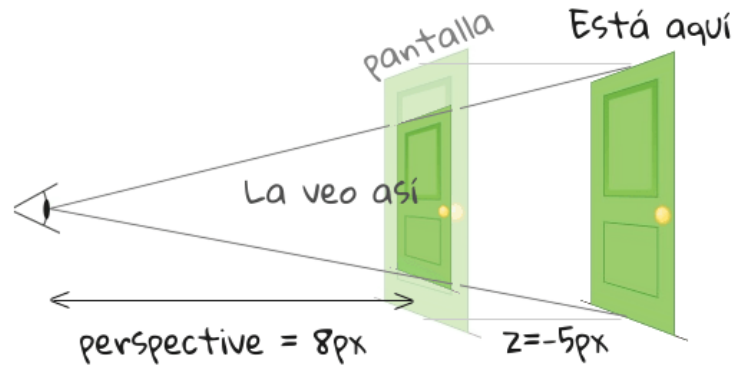


Pero si queremos auténtica perspectiva tenemos que considerar algunos detalles, como la **distancia de cada capa** al observador. Esta distancia la vamos a dar como una distancia medida sobre el eje Z (el eje perpendicular a la pantalla), o sea, una coordenada que puede ser positiva o negativa si alejamos el objeto.



Si a un objeto le ponemos una Z positivo lo estamos acercando y si le ponemos un valor negativo lo estamos alejando del punto donde situamos al observador, o sea el ojo.

La distancia entre el ojo y el centro de la pantalla es lo que vamos a llamar perspectiva, por su nombre CSS: **perspective**. Esta imagen te aclarará la idea. La puerta está al fondo pero yo la veo en la pantalla más pequeña. Si estuviera por delante ocurre al revés: estando delante de la pantalla al proyectarla sobre esta la veré más grande.



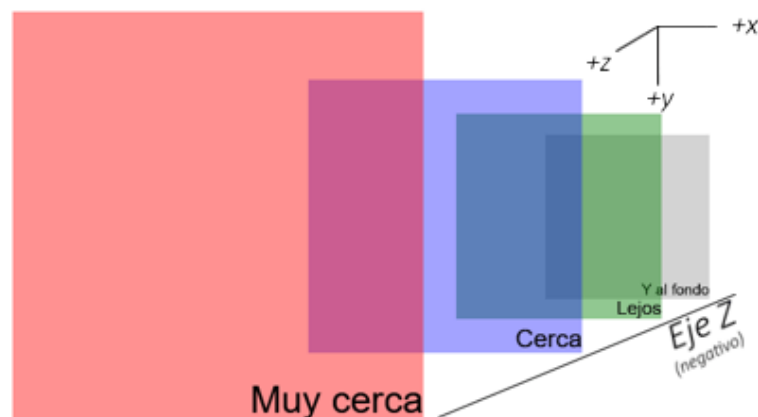
Es necesario considerar que al alejar el objeto su **tamaño y posición relativa cambian** aparentemente al situarlos a una cierta distancia del observador. El valor de escalado del tamaño viene dado por una formulita muy sencilla

$$\text{escala} = 1 - \frac{Z}{\text{perspectiva}}$$

También las posiciones vertical y horizontal se modifican, pero esta vez la formulita es más compleja. Depende del ancho de la pantalla, del tamaño del objeto, de la posición y de la escala. No te asustes, es probable que no le necesites nunca:

$$\frac{(\text{pantalla} - \text{objeto} - 2 * \text{posicion}) * Z}{2 * (\text{persp} - z)}$$

En esta formulita pantalla y objeto son los tamaños ancho o alto dependiendo si calculamos la posición: horizontal o vertical. Lógicamente si los objetos están centrados en la pantalla, la posición no se modifica al situarlos más o menos alejados del observador.



Los tamaños y posiciones parecen cambiar al alejarlos

Este es el ejemplo que podrás crear si usas el código que te pongo en el siguiente apartado. Aquí los objetos están a distintas distancias del observador y además no van a estar centrados, para observar bien el efecto parallax.

## Una muestra gráfica

Para ver la idea del parallax en la web puedes probar el siguiente ejemplo. Vas a ver la imagen de arriba: distintos cuadros con colores de fondo colocados uno tras otro, a distintas distancias de la pantalla. Además no están centrados, están colocados a la izquierda del centro de la pantalla. Tienes una perspectiva total de la escena.

Primero te pongo los estilos CSS y luego el código de la página

```
<style>
html{
  height:100%;
  overflow-x:hidden;
  font-family:sans-serif;
}
body{
  overflow-y:scroll;
  margin:0;
  height:100%;
  transform-style: preserve-3d;
  perspective: 2px;
}
.contenedor{
  position:relative;
  display:flex;
  align-items: center;
  min-height: 100vh;
  width:100%;
  transform-style: inherit;
}
.contenedor div{
  position:absolute;
  height:350px;
  width:350px;
  left:100px;
  font-size: 2rem;
  display:flex;
  justify-content:flex-end;
  align-items:flex-end;
}
#fondo1{
  background-color: #ff0000e;
  transform: translateZ(0px);
  z-index: -1;
}
#fondo2{
```

```

background-color: #0000ff5c;
transform: translateZ(-1px);
z-index: -2;
}
#fondo3{
background-color: #0080006e;
transform: translateZ(-2px);
z-index: -3;
}
#fondo4{
background-color: lightgray;
transform: translateZ(-3px);
z-index: -4;
}
.relleno{height:50vh}

</style>

```

El estilo CSS define los parámetros fundamentales para cada capa y para el punto del observador. En este ejemplo el contenedor principal para el scroll va a ser el elemento body. Le ponemos la propiedad **perspective** a 2px, sería como la distancia desde la que se ve el plano del dibujo.

También es necesario ajustar la propiedad **transform-style: preserve-3d**, para que mantenga el contexto 3D. En Chrome esto no afecta, pero es fundamental para Mozilla.

Cada capa (los bloques fondo1 a fondo4) se sitúa sobre el eje Z mediante **translateZ()**, el argumento indica la posición en la que se coloca cada fondo. Dejamos que el navegador calcule el tamaño y posición de cada uno.

Vamos a dejar que se vean en el tamaño natural, no compensamos el escalado.

```

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Ejemplo Parallax</title>
</head>
<body>
  <div class="contenedor">
    <div id="fondo1">Muy cerca</div>
    <div id="fondo2">Cerca</div>
    <div id="fondo3">Lejos</div>
    <div id="fondo4">Y al fondo</div>
  </div>
  <div class="relleno"/>
</body>
</html>

```

Si pruebas el ejemplo en pantalla grande lo verás aún mejor. Pero queda clara la idea: los cuadros cercanos quedan como más grandes que los lejanos. Tenemos

una perspectiva gracias al parallax. Como están situados a un lado de la pantalla, la posición también se ve afectada por el alejamiento.

Como ves cada cuadro se escala a medida que se aleja de nosotros.

Esta escala tiene un valor que viene dado por **escala = (1 - posZ/presp)**, donde **posZ** es la distancia con su signo en el eje Z (`translateZ(z)`) mientras que **presp** es el valor de **perspective** del cuadro contenedor (en este caso es `body`).

Si quieres mantener los tamaños de cada cuadro aplicas la propiedad **transform: scale(escala)**. Por ejemplo el cuadro *fondo4* tendría un valor de escala igual a **(1 - (-3/2))** que es **2.5**. Le pondrías la propiedad **transform: translateZ(-3px) scale(2.5)**, con lo que lo verás a su ancho real. Pero el efecto parallax no se perderá, su movimiento seguirá siendo relativo a su posición: más lento que el cuadro en primer plano.

Si quieres ver todos los bloques centrados en medio de la pantalla cambia la regla del selector **.contenedor div**, pon la propiedad **left** a l centro de la pantalla, para eso usa la función `calc` **left: calc(50% - 175px)**; (175px es la mitad del ancho de los bloques).



Los bloques están centrados y el efecto parallax hace que los distintos cuadros vayan moviéndose a distintas velocidades, los más profundos son los más lentos.