

Propiedades CSS3

CSS3 se ha dividido en "módulos". Contiene la "antigua especificación CSS" (que se ha dividido en piezas más pequeñas). Además, se añaden nuevos módulos.

Bordes redondeados

Los bordes podrán ser redondos con posibilidad de indicar el radio de curvatura. Si especifica sólo un valor para la propiedad border-radius, este radio se aplicará a las 4 esquinas. Sin embargo, puede especificar cada esquina por separado si lo desea:

Cuatro valores: el primer valor se aplica a la parte superior izquierda, el segundo se aplica a la parte superior derecha, el tercer valor se aplica a la parte inferior derecha y el cuarto se aplica a la esquina inferior izquierda

Tres valores: el primer valor se aplica a la parte superior izquierda, el segundo se aplica a la parte superior derecha y la parte inferior izquierda, y el tercer valor se aplica a la parte inferior derecha

Dos valores: el primer valor se aplica a la esquina superior izquierda ya la esquina inferior derecha, y el segundo se aplica a la esquina superior derecha y a la esquina inferior izquierda.

```
#rcorners4 {
  border-radius: 15px 50px 30px 5px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
#rcorners5 {
  border-radius: 15px 50px 30px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
#rcorners6 {
  border-radius: 15px 50px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
<p>Four values - border-radius: 15px 50px 30px 5px:</p>
<p id="rcorners4"></p>
<p>Three values - border-radius: 15px 50px 30px:</p>
<p id="rcorners5"></p>
<p>Two values - border-radius: 15px 50px:</p>
<p id="rcorners6"></p>
```

Cuatro valores - border-radius: 15px 50px 30px 5px:



Tres valores - border-radius: 15px 50px 30px:



Dos valores - border-radius: 15px 50px:



Imágenes en los bordes

CSS3 Presento la forma en que podemos añadir imágenes a los bordes de los elementos HTML por medio de la propiedad `border-image` que se une a la propiedad clásica `border` para extender la forma en que damos estilos a los bordes de los elemento.

`Border-image: --valores--;`

En realidad `border-image` es el método shorthand de las propiedades:

- `border-image-source`
- `border-image-slice`
- `border-image-width`
- `border-image-ouset`
- `border-image-repeat`

y en resumen la sintaxis final quedaría de la siguiente forma:

`border-image: source slice width outset repeat;`

Así que vamos a ver cada una de las propiedades que forman a `border-image`.

`border-image-source`

Esta propiedad la utilizamos para establecer la ruta de la imagen que vamos a establecer como fondo del borde, puede ser una ruta absoluta o relativa. Utilizamos la función `url()` y como argumento colocamos la ruta específica. O también podemos colocar el valor `none`, y se mostrará el borde del elemento por defecto.

```
#elemento{  
    border-image-source: url(imagen-borde.png);  
}
```

Otra cosa también que debemos de tener en cuenta es que tenemos que definir la propiedad `borde` como base de la propiedad `border-image`.

Ejemplo: `border: 30px solid;`

```
#elemento {  
    margin: 100px auto;  
    height: 150px;  
    width: 150px;  
    border: 30px solid;  
    border-image-source: url(http://tutosytips.com/wp-content/ejemplos/border-image.png);  
}
```

Si solo definimos `border-image-source` como en el código CSS anterior obtendremos:



border-image-slice

```
#elemento{  
  border-image-slice: /* Valor */;  
}
```

Lo que hace esta propiedad es dividir o cortar la imagen que especificamos en `border-image-source` en 9 partes. En las 4 esquinas de la imagen, los 4 lados y la parte central de la imagen.

En esta propiedad podemos especificar como valores:

Números: Representa píxeles para imágenes de mapa de bits y para las imágenes vectoriales coordenadas.

Porcentajes: Los valores de los porcentuales son respecto a la altura o la anchura de la imagen.

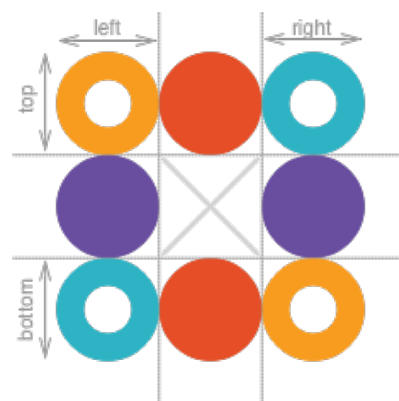
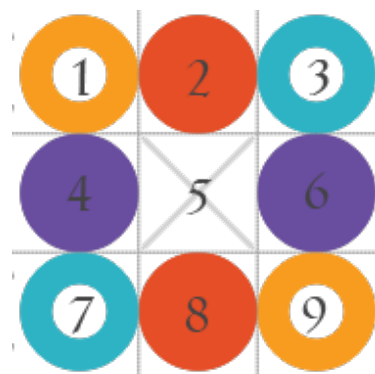
Fill: Esta palabra clave obliga al uso de la corte de la imagen central para que se muestre como fondo del elemento.

Inherit: Es una palabra clave que indica que los cuatro valores se heredan de elemento calculado del valor de sus padres.

Una vez definamos el valor a utilizar podemos utilizar Hasta cuatro valores diferentes se pueden especificar, en el siguiente orden: top, right, bottom, left.

Al igual que en los bordes podemos usar de 1 a 4 valores. Para el siguiente código:

```
#elemento {  
  margin: 100px auto;  
  height: 150px;  
  width: 150px;  
  border: 30px solid;  
  border-image-source: url(border-image.png);  
  border-image-slice: 30 20 30 20;  
}
```



border-image-width

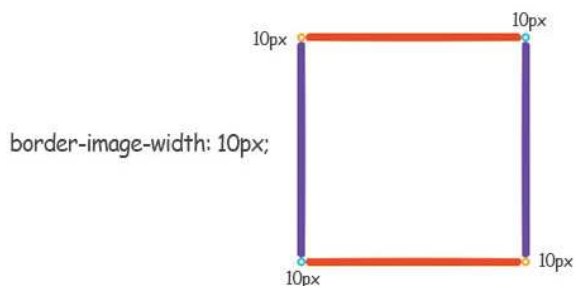
```
border-image-width: /*valor*/ ;
```

Con `border-image-width` determinamos el ancho de la imagen que se está aplicando al borde del elemento. El valor de esta propiedad lo podemos especificar en cualquier medida de longitud ya sea relativa o absoluta, si utilizamos porcentajes (%) el valor se aplicara a la anchura y altura de la zona del borde

Propiedades CSS3

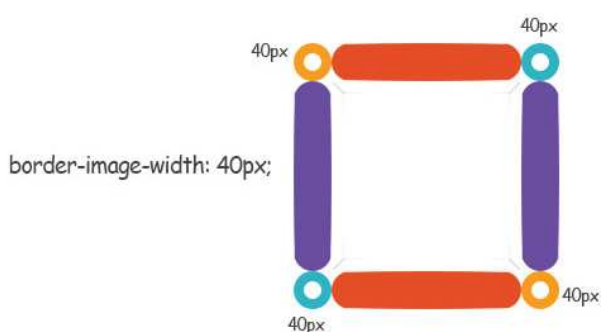
cortada con `border-image-slice`, no podemos usar valores negativos y el valor inicial es none.

También podemos usar de 1 a cuatro valores como con `slice` y cada valor se aplicara a cada lado.



`border-image-width: 10px;`

Este valor en `border-image-width` no es del todo dependiente del valor de la propiedad `border-width`, ya que el primero es para definir el grosor del borde y el otro para definir el tamaño de la imagen.



`border-image-width: 40px;`

border-image-repeat

`border-image-repeat: /*Valor*/ ;`

`Border-image-repeat` nos va a permitir configurar la forma en que se deben de mostrar los cortes hechos por `slice` en la imagen que estamos utilizando como del borde de un elemento. Este es el valor inicial de esta propiedad.

Los valores que podemos utilizar son:

Stretch: que hace estirar la imagen para ocupar y llenar todo el espacio del borde.

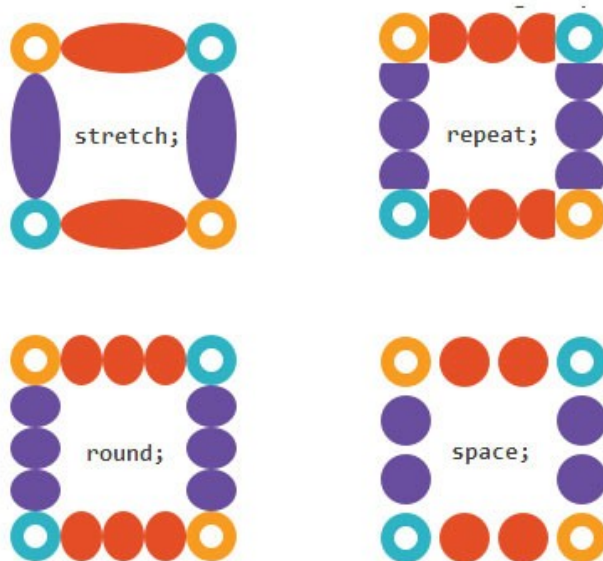
Repeat: va a repetir los cortes para llenar y ocupar todo el espacio del borde.

Round: con este valor lo que hacemos es que los cortes de la imagen del borde se distribuyan en el espacio que le genera el borde llenándolo todo.

Space: hace casi igual lo que hace `round`, solo que esta vez lo que se va a distribuir para llenar el borde son los espacios y no los cortes de la imagen.

Veamos los diferentes comportamientos de los valores de esta propiedad.

Propiedades CSS3



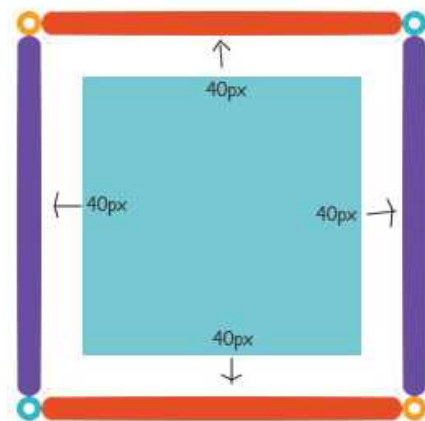
border-image-outset

`border-image-outset: /* valor */ ;`

Esta propiedad nos ayuda a especificar hasta que punto fuera del elemento y del borde del elemento padre se debe de colocar la imagen aplicada en el `border-image-source`. Es decir cuánto espacio hacia afuera debe de haber entre el elemento y la imagen del borde.

Podemos indicar cualquier valor ya sea relativo o absoluto.

```
#elemento {  
    margin: 100px auto;  
    height: 150px;  
    width: 150px;  
    border: 10px solid;  
    border-image-source: url(border-image.png);  
    border-image-slice: 40;  
    border-image-width: 20px;  
    border-image-outset: 40px;  
    background: #75c8d3;  
}
```



Seguimos con el mismo ejemplo y esta vez le decimos que aleje el `border-image` 40px desde su lugar inicial en su elemento padre con la propiedad `border-image-outset: 40px;`

Cabe decir que también podemos ajustar cada lado individualmente.

Propiedades CSS3

border-image (Shorthand)

Y por último, decir que podemos reunir todas las propiedades anteriores en una sola declaración, y esto lo hacemos dentro de la propiedad `border-image`;

La propiedad abreviada:

```
border-image: border-image-source border-image-slice border-imagewidth border-image-outset border-image-repeat;
```

Cuando cualquiera de los valores anteriores se omiten, se utiliza el valor por defecto.

Veamos un ejemplo de todas las propiedades juntas en una declaración:

```
#elemento {  
    margin: 100px auto;  
    height: 150px;  
    width: 150px;  
    border: 10px solid;  
    background: #75c8d3;  
    border-image: url(http://tutosytips.com/wp-content/ejemplos/border-image.png) 40 / 20px / 40px stretch;  
}
```

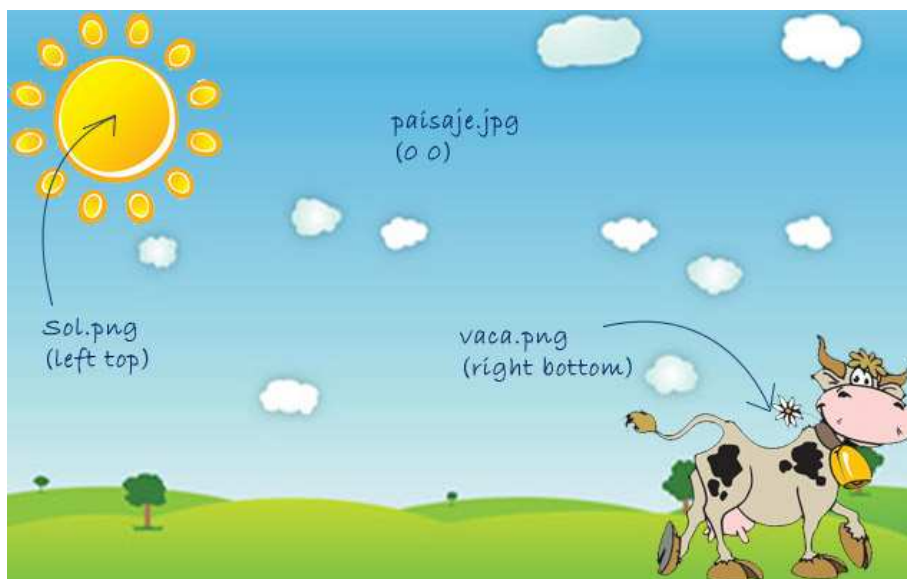


Fondos:

background-image

Entre las principales características destacan que se podrá agregar múltiples fondos a un objeto. La declaración de la propiedad es la misma de la anterior versión de css, solo que ahora añadimos múltiples imágenes a la propiedad `background-image` y la separamos por medio de comas.

```
div{  
  
height:400px;  
  
width: 635px;  
background-image:  
url(sol.png),  
url(vaca.png),  
url(paisaje.jpg);  
background-repeat: no-repeat;  
background-position:  
left top,  
right bottom,  
0 0;}
```



background-size

Background-size es otra propiedad de CSS3 que viene con el título de interesante, por medio de esta propiedad vamos a poder ajustar o escalar nuestro background o imagen de fondo de nuestro sitio web, podemos ajustar el ancho y alto, podemos ajustarla y adaptarla a la resolución de la pantalla del navegador o div contenedor.

```
background-size: 400px 635px;
```

En el caso anterior estamos aplicando un tamaño de 400px de ancho, y 635px de alto.

El tamaño también lo podemos definir por medio de porcentaje, unidad de medida, cover y contain.

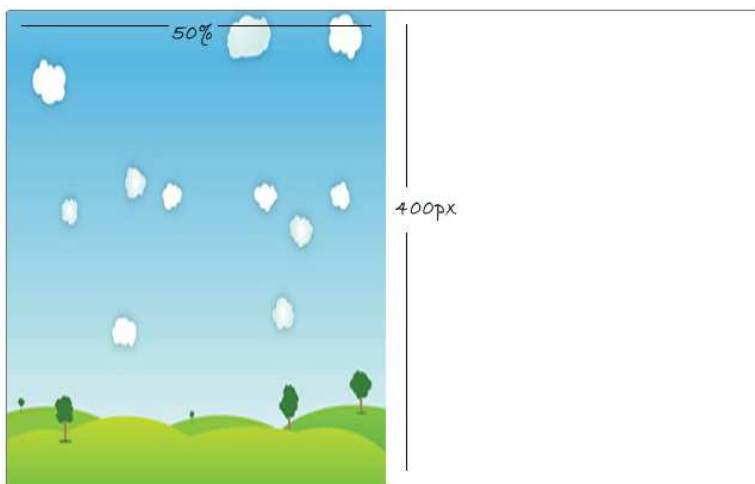
Porcentaje: Establecemos el tamaño de la imagen dependiendo de su del elemento padre.

Cover: Escala la imagen para el tamaño más pequeño de tal manera que su anchura y su altura puede caber dentro del área de contenido.

Contain: Escala la imagen al tamaño más grande de tal manera que su anchura y su altura puede caber dentro del área de contenido.

```
div{  
    height:400px;  
    width: 635px;  
    background:url(paisaje.jpg)  
    0 0 no-repeat;  
    background-size: 50% 400px;  
    border: 1px solid #777;  
}
```

Con esa regla el resultado seria el siguiente:



background-size: contain;



background-size: cover;



Propiedades CSS3

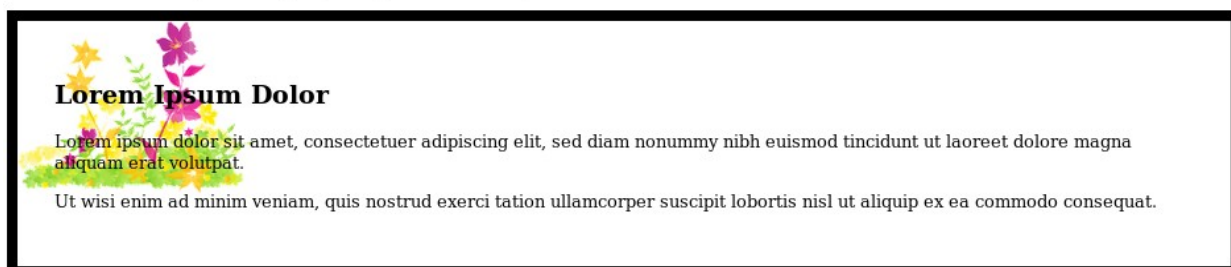
Utilizando el 100% del tamaño de la imagen vamos a tener la opción de que la imagen se redimensione y se adapte al navegador o a la caja que contenga la imagen.

Background-origin

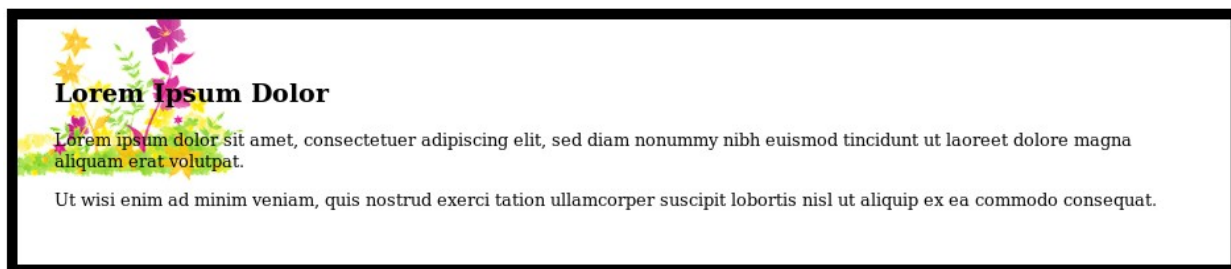
Esta propiedad nos permite posicionar el background o imagen de fondo en relación al contenido o área de la caja contenedora, podemos hacer que sea relativa al padding de la caja por medio de padding-box, o al borde de la caja por medio de border-box y también la podemos ubicar relativa a el contenido de la caja por medio de content-box.

<pre>#example1 { border: 10px solid black; padding: 35px; background: url(img_flwr.gif); background-repeat: no-repeat; }</pre>	<pre>#example2 { border: 10px solid black; padding: 35px; background: url(img_flwr.gif); background-repeat: no-repeat; background-origin: border-box; }</pre>	<pre>#example3 { border: 10px solid black; padding: 35px; background: url(img_flwr.gif); background-repeat: no-repeat; background-origin: content-box; }</pre>
--	---	--

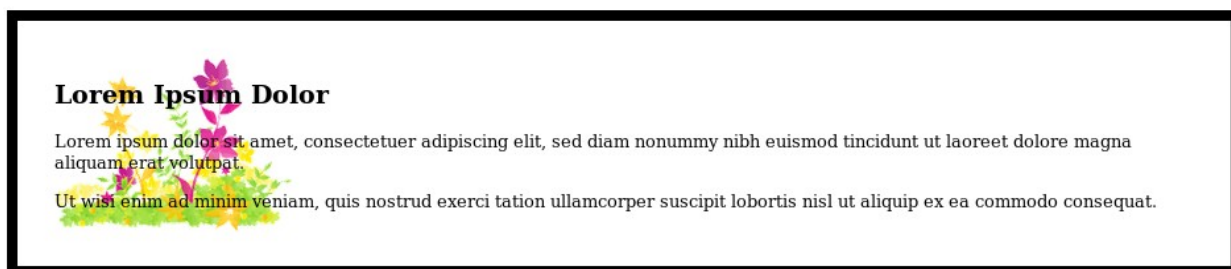
No se establece background-origin (padding-box es el valor por defecto):



background-origin: border-box:



background-origin: content-box:



Propiedades CSS3

Background-clip

La propiedad CSS3 background-clip especifica el área de dibujo del fondo. La propiedad puede tomar tres valores diferentes:

Border-box - (por defecto) el fondo se pinta al borde exterior del borde

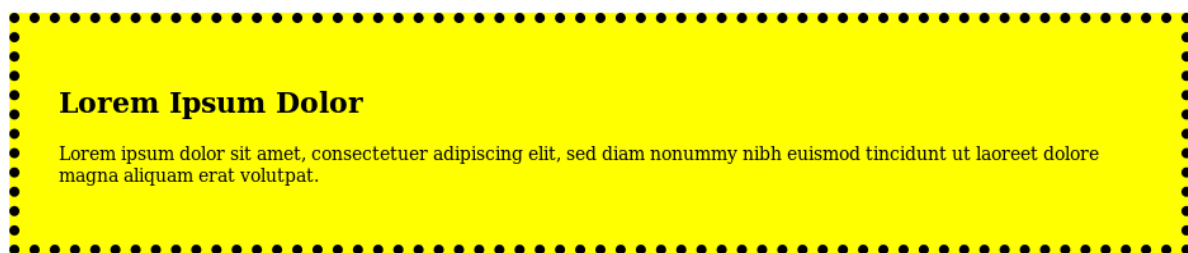
Padding-box - el fondo se pinta al borde exterior del relleno

Content-box - el fondo se pinta dentro del cuadro de contenido

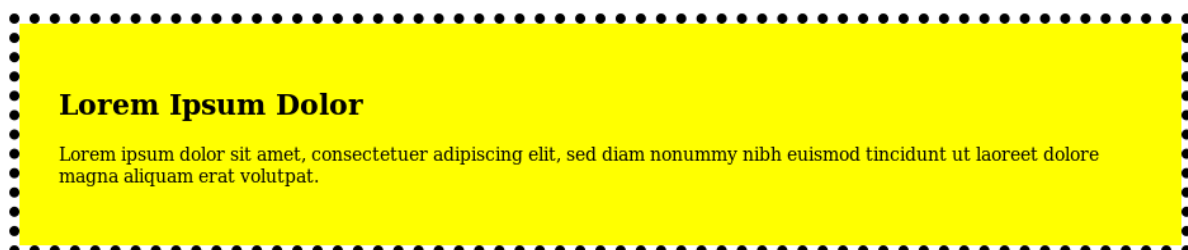
Los siguientes estilos sobre tres divs:

<pre>#example1 { border: 10px dotted black; padding:35px; background: yellow; }</pre>	<pre>#example2 { border: 10px dotted black; padding:35px; background: yellow; background-clip: padding-box; }</pre>	<pre>#example3 { border: 10px dotted black; padding:35px; background: yellow; background-clip: content-box; }</pre>
---	---	---

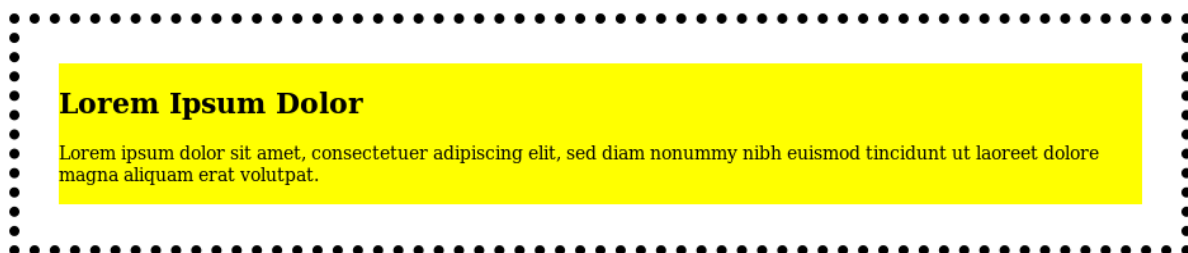
No se define background-clip (border-box es el valor por defecto):



background-clip: padding-box:



background-clip: content-box:



Propiedades CSS3

Colores

CSS admite nombres de color, hexadecimal y colores RGB.

Además, CSS3 también introduce:

Colores RGBA `#p1 {background-color: rgba(255, 0, 0, 0.3);} /* rojo con opacidad */`

Colores HSL `#p1 {background-color: hsl(120, 100%, 50%);} /* green */`

Colores HSLA `#p3 {background-color: hsla(120, 100%, 25%, 0.3);} /* dark green with opacity */`

Opacidad `#p3 {background-color: rgb(0,0,255);opacity:0.6;} /* blue with opacity */`

Gradientes

Esta propiedad permite definir fondo con gradientes, lineales y radiales. Para ello necesitaremos definir una dirección, un color y luego diferentes "paradas" que serán los colores hacia los que se harán los gradientes:

Ejemplos lineales:

* de arriba hacia abajo:

```
#grad {
    background: red; /* para navegadores que no lo soportan */
    background: -webkit-linear-gradient(red, yellow); /* For Safari 5.1 to 6.0 */
    /*
    background: -o-linear-gradient(red, yellow); /* For Opera 11.1 to 12.0 */
    background: -moz-linear-gradient(red, yellow); /* For Firefox 3.6 to 15 */
    background: linear-gradient(red, yellow); /* Standard syntax */
}
```



* de izquierda a derecha:

```
#grad {
    background: red; /* para navegadores que no lo soportan */
    background: -webkit-linear-gradient(left, red , yellow); /* For Safari 5.1 to 6.0 */
    /*
    background: -o-linear-gradient(right, red, yellow); /* For Opera 11.1 to 12.0 */
    background: -moz-linear-gradient(right, red, yellow); /* For Firefox 3.6 to 15 */
    /*
    background: linear-gradient(to right, red , yellow); /* Standard syntax */
}
```



Es posible obtener más variaciones como diagonal.

```
background: linear-gradient(left top, red, yellow);
```

usar ángulos

```
background: linear-gradient(-90deg, red, yellow);
```

múltiples paradas

```
background: linear-gradient(red, yellow, green);
```

transparencias

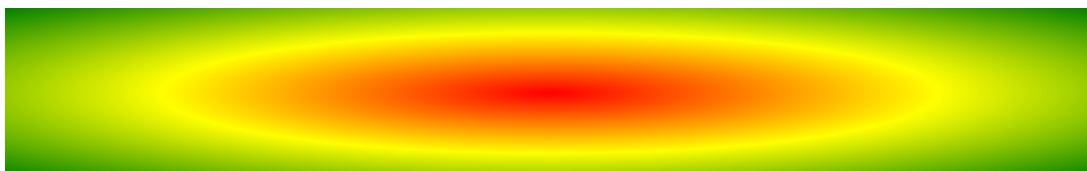
```
background: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));
```

repetir gradientes lineales

```
background: repeating-linear-gradient(red, yellow 10%, green 20%);
```

Ejemplos radiales:

```
#grad {  
  background: red; /* para navegadores que no lo soportan */  
  background: -webkit-radial-gradient(red, yellow, green); /* Safari 5.1 to 6.0 */  
  background: -o-radial-gradient(red, yellow, green); /* For Opera 11.6 to 12.0 */  
  background: -moz-radial-gradient(red, yellow, green); /* For Firefox 3.6 to 15 */  
  background: radial-gradient(red, yellow, green); /* Standard syntax */  
}
```



También podemos repetir gradientes radiales

```
background: repeating-radial-gradient(red, yellow 10%, green 15%);
```

Y existen más opciones como usar palabras clave y otras.

Sombras

Podemos definir sombras, tanto para texto, text-shadow como para cajas, box-shadow.

Propiedades CSS3

text-shadow

El primer valor define el valor horizontal y el segundo el vertical. El tercer valor corresponde al difuminado, y el cuarto al color. Estos valores son optativos.

```
h1 {  
    color: white;  
    text-shadow: 2px 2px 4px  
#000000;  
}
```

Text shadow effect!

Y con múltiple sombras:

```
h1 {  
    color: white;  
    text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;  
}
```

box-shadow

De igual manera que con text-shadow

```
div {  
    box-shadow: 10px 10px 5px grey;  
}
```

This is a yellow <div> element with
a blurred, grey box-shadow

Texto

CSS3 contiene varias nuevas características:

- Desbordamiento de texto
- Ajuste de línea
- Ruptura de palabras

La propiedad de desbordamiento de texto CSS3 especifica cómo se debe señalar al usuario el contenido desbordado que no se muestra (se ha de acompañar con `overflow:hidden`).

Se puede recortar: `text-overflow: clip`;

O se puede representar como una elipsis (...): `text-overflow: ellipsis`;

Fuentes

Con css podemos importar fuentes y renombrarlas para un uso más semántico. Para ello deberemos importar la fuente.

```
@font-face {  
    font-family: myFirstFont;  
    src: url(sansation_light.woff);  
}  
div { font-family: myFirstFont; }
```

Residimensionar

La propiedad `resize` especifica si el usuario debe o no redimensionar un elemento. Puede usar los valores `horizontal`, `vertical` o `both`:

```
div {  
    resize: horizontal;  
    overflow: auto;  
}
```

Contornos

La propiedad `outline-offset` agrega espacio entre el contorno y el borde de un elemento.

Los contornos difieren de las bordes:

- Un contorno es una línea dibujada alrededor de elementos, por fuera del borde.
- Un contorno no ocupa espacio
- Un contorno puede ser no rectangular

```
div {  
    border: 1px solid black;  
    outline: 1px solid red;  
    outline-offset: 15px;  
}
```

Estructura multi-columna

Ahora será más sencillo crear diseños con múltiples columnas sin etiquetas adicionales. En el primer ejemplo, vemos la implementación de esta opción en la versión actual de CSS. Para este caso, debemos crear dos clases CSS y dos objetos DIV para simular la multi-columna.

```
<div class="entry-content">  
    <div class="col first">    </div>  
    <div class="col">    </div>  
</div>  
  
.entry-content .col {  
    float: left;  
    width: 467px;  
    margin-left: 30px;  
    height: 300px;  
    background:gray; }
```

Propiedades CSS3

```
.entry-content .first { margin-left: 0;}  
.col{ height: 300px; background: gray;}
```

Y en CSS 3 la definición es más sencilla y requiere menos procesamiento

```
<div class="entry-content" content"></div>  
  
.entry-content {  
    column-count: 2;  
    column-gap: 30px;  
}
```

La propiedad de column-gap especifica la separación entre las columnas.

Si queremos una barra de separación usaremos la propiedad column-rule-style. Esta puede estilarse con column-rule-color y column-rule-width. También disponemos de una propiedad abreviada column-rule.

El siguiente ejemplo coge el contenido de un div.newspaper y lo 'reparte' en tres columnas:

```
.newspaper {  
    -webkit-column-count: 3; /* Chrome, Safari, Opera */  
    -moz-column-count: 3; /* Firefox */  
    column-count: 3;  
    -webkit-column-gap: 40px; /* Chrome, Safari, Opera */  
    -moz-column-gap: 40px; /* Firefox */  
    column-gap: 40px;  
    -webkit-column-rule: 1px solid lightblue; /* Chrome, Safari, Opera */  
    -moz-column-rule: 1px solid lightblue; /* Firefox */  
    column-rule: 1px solid lightblue;  
}
```

Transiciones

Con esta nueva propiedad de CSS3 vamos a poder aplicar efectos de transiciones para cambiar de un estilo a otro sin tener que utilizar la tecnología flash o Javascript, solamente CSS3.

Primero tienes que saber que esta propiedad no es acogida por los navegadores de una manera estándar, si no que vamos a tener que utilizar los prefijos de cada navegador.

Propiedades CSS3

- * Firefox 4+ requiere el prefijo -moz-.
- * Chrome y Safari requiere el prefijo -webkit-.
- * Opera requiere que el prefijo -o-.

Transition y sus propiedades

Segun la [W3 y el modulo que trabaja en transition](#) existen 4 propiedades hasta ahora: (transition-property, transition-duration, transition-timing-function, y transition-delay.)

transition-property: especifica el nombre de la propiedad CSS a la que se aplica la transición.

transition-duration: define el tiempo que tomara la transición.

transition-timing-function: Especifica la curva de velocidad del efecto de la transición

transition-delay: determina el tiempo que transcurrirá antes de comenzar la transición.

transition-property – (propiedad de la transición):

Con la propiedad transition-property declaramos el nombre de la propiedad CSS sobre la que se aplica la transición. Los valores posibles son: all o none y por supuesto la propiedad o propiedades.

all = Todas las propiedades tendrán un efecto de transición.

none = Ninguna de las propiedades tendrán un efecto de transición.

transition-property:height, border-radius, box-shadow;

la utilizamos para listar las propiedades que se cambiarán, ya sea una o varias separadas por comas.

transition-duration – (duración de la transición):

La propiedad de transition-duration o duración de la transición especifica cuántos segundos (s) o milisegundos (ms) tendrá un efecto de transición para llegar al final.

transition-duration: 5s, 500ms;

transition-timing-function – (temporización de la transición):

Esta propiedad es la que determina cómo se desarrolla la transición en el tiempo estipulado, es decir cual será su comportamiento en el desarrollo de la

Propiedades CSS3

transición. Esta propiedad cuenta con varios valores de comportamiento, que son los siguientes:

- **linear**: Especifica un efecto de transición con la misma velocidad de principio a fin.
- **ease**: Especifica un efecto de transición con un comienzo lento, luego rápido, luego terminan lentamente
- **ease-in**: Especifica un efecto de transición con un comienzo lento
- **ease-out**: Especifica un efecto de transición con un final lento
- **ease-in-out**: Especifica un efecto de transición con un comienzo lento y final lento.
- **cubic-bezier(n,n,n,n)**: Nos hace Definir valores propios de principio a fin. Los valores posibles son los valores numéricos de 0 a 1

Veamos un ejemplo de cada uno de los valores donde aplicamos a cada barra un valor diferente, Claro que debemos asignarle los prefijos de cada navegador

```
#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}
```

transition-delay – (demora de la transición):

Especifica el retraso con el que comienza la transición. El valor de retardo de transición se define en segundos (s) o milisegundos (ms) y admite valores negativos y positivos:

la sintaxis es : transition-delay: time;

Veamos un Ejemplo con varios valores:

```
transition-property:width; /* se aplica al whidth */
transition-duration:5s; /* duracion de la transición */
transition-delay:2s; /* retraso de la transición */
```

Lo que hace la anterior función es hacer la transición después de 2 segundo de estar en el estado hover, y despues de 2 segundos vuelve al estado original.

La propiedad transition resumida:

Estas propiedades también tienen una forma acortada o resumida para declarar todas o algunas de ellas lo que llamamos (Shorthand), que lo que hacemos es combinar las 4 propiedades, que seria algo así.

Propiedades CSS3

transition: property duration timing-function delay;

transicion: [<transition-property> | | <transition-duration> | | <transition-timing-function> | | <transition-delay>]

Mas informacion en http://www.w3schools.com/cssref/css3_pr_transition.asp

Animaciones

Las animaciones CSS3 permiten la animación de la mayoría de los elementos HTML sin usar JavaScript o Flash!

¿Qué son las animaciones CSS3?

Una animación permite que un elemento cambie gradualmente de un estilo a otro.

Puede cambiar la cantidad de propiedades CSS que desee, tantas veces como desee.

Para utilizar la animación CSS3, primero debe especificar algunos fotogramas clave para la animación.

Los fotogramas clave mantienen los estilos que el elemento tendrá en determinados momentos.

La regla @keyframes

Cuando se especifican estilos CSS dentro de la regla @keyframes, la animación cambiará gradualmente del estilo actual al nuevo estilo en determinados momentos.

Para que una animación funcione, debe vincular la animación a un elemento.

El siguiente ejemplo vincula la animación "ejemplo" al elemento <div>. La animación durará 4 segundos y cambiará gradualmente el color de fondo del elemento <div> de "rojo" a "amarillo":

```
/* The animation code */
@keyframes ejemplo {
  from {background-color: red;}
  to {background-color: yellow;}
}
/* El elemento al que se le aplica la animación */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: ejemplo;
  animation-duration: 4s;
}
```

Nota: Si la propiedad animation-duration no está especificada, la animación no tendrá ningún efecto, ya que el valor predeterminado es 0.

Propiedades CSS3

En el ejemplo anterior hemos especificado cuando el estilo cambiará utilizando las palabras clave "from (de)" y "to (a)" (que representa 0% (inicio) y 100% (completado)).

También es posible utilizar el porcentaje. Mediante el uso de porcentaje, puede agregar tantos cambios de estilo como desee.

El siguiente ejemplo cambiará el color de fondo del elemento <div> cuando la animación esté completada al 25%, al 50%, y de nuevo cuando la animación esté completa al 100%:

```
/* El código de la animación */
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}

/* El elemento al que se aplica la animación */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s; }
```

Retrasar una animación

La propiedad animation-delay especifica un retardo para el inicio de una animación.

Para aplicar un retraso de 2seg al ejemplo anterior, añadir animation-delay: 2s;

Definir cuántas veces debería ejecutarse una animación

La propiedad animation-iteration-count especifica el número de veces que debe ejecutarse una animación y además este valor puede ser infinito (infinite). Por ejemplo:

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

Ejecutar animación en sentido inverso o ciclos alternativos

La propiedad animation-direction se utiliza para permitir que una animación se ejecute en dirección inversa o ciclos alternos. Por ejemplo:

Propiedades CSS3

Ejecutar la animación en sentido inverso `animation-direction: reverse;`

Hacer que la animación se ejecute primero hacia adelante, luego hacia atrás y luego hacia adelante `animation-direction: alternate;`

Especificar la curva de velocidad de la animación

La propiedad `animation-timing-function` especifica la curva de velocidad de la animación.

La propiedad `animation-timing-function` puede tener los mismos valores que `transition-timing-function`.

El siguiente ejemplo muestra algunas de las diferentes curvas de velocidad que se pueden usar:

```
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
```

Propiedad abreviada

El siguiente ejemplo utiliza seis de las propiedades de animación:

```
div {
  animation-name: example;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

El mismo efecto de animación que el anterior se puede lograr utilizando la propiedad de animación abreviada:

```
div {
  animation: example 5s linear 2s infinite alternate;
}
```

Imágenes

Una muestra de lo que se puede hacer con imágenes con un poco de ingenio.

Una imagen como link y aplicando una transición al borde durante el hover:

```
a {
    display: inline-block;
    border: 1px solid #ddd;
    border-radius: 4px;
    padding: 5px;
    transition: 0.3s;
}
a:hover {
    box-shadow: 0 0 2px 1px rgba(0, 140, 186, 0.5);
}
<a href="paris.jpg">  </a>
```

Imágenes responsivas:

```
img { max-width: 100%; height: auto; }
```

Filtros en imágenes:

La propiedad de filtro CSS agrega efectos visuales (como desenfoque y saturación) a un elemento.

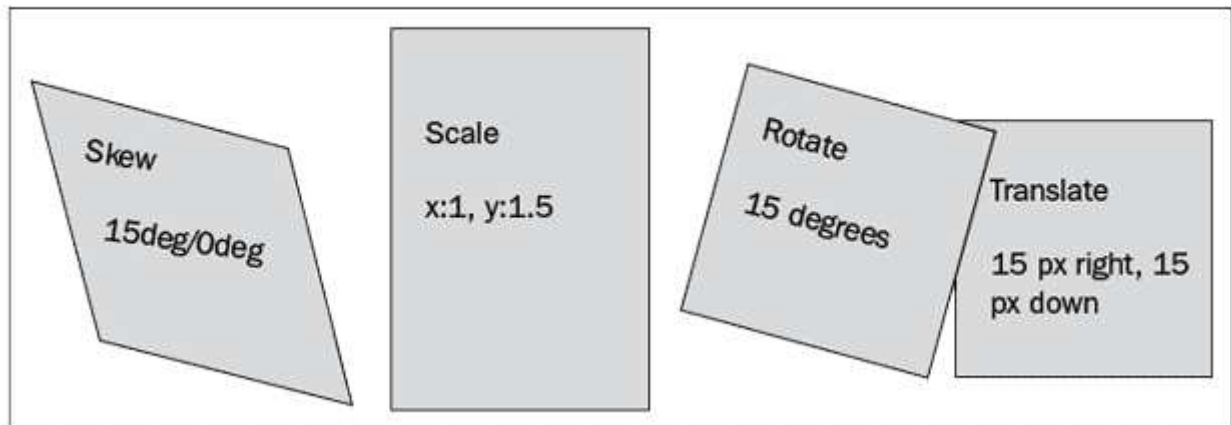
Ejemplo, cambiar el color de todas las imágenes a blanco y negro (100% gris):

```
img {
    -webkit-filter: grayscale(100%); /* Safari 6.0 - 9.0 */
    filter: grayscale(100%);
}
```

Transformaciones 2D

La propiedad Transform en CSS3 nos va a permitir hacer cambios a los objetos del documento al que le apliquemos este estilo, esta propiedad nos va a permitir hacer las siguientes transformaciones a los elementos: translate (traducir), rotate (rotar), scale (escalar) o skew (sesgar), efectos que antes era imposible hacer con CSS y para usarlos en la web teníamos que utilizar flash o javascript. Bueno damos gracias que ya CSS3 podemos hacer estas funciones. Bueno antes de comenzar miremos el soporte de cada navegador sobre esta propiedad.

Transformaciones Básicas con CSS3



translate

o Transformación de Posición:

Esta propiedad de CSS3 nos permite mover elementos a la izquierda, derecha, arriba o abajo mediante la función translation, esta función es similar al comportamiento de position que trabajábamos en css2: position: relative; donde se declaraba su posición en superior e izquierdo por ejemplo. Cuando se emplea una función de Translation a un objeto lo podemos mover sin afectar el flujo de trabajo del documento.

Un elemento con la transformación translation puede moverse con relación a su posición actual.

Esta transformación se mueve en función (x,y) es decir x de la izquierda a derecha - y de la parte superior a inferior

transform: translate(x,y);

Para mover el el siguiente h1 45px desde la izquierda y 45px desde la parte superior del div :<div> <h1>Transform - Translate</h1> </div>

```
div {
  background: #ccc;
  box-shadow: 2px 2px 3px #777;
  width: 350px;
}
h1 {
  -webkit-transform: translate(45px,45px);
  -moz-transform: translate(45px,45px);
  -ms-transform: translate(45px,45px);
  -o-transform: translate(45px,45px);
  transform: translate(45px,45px);
}
```



Propiedades CSS3

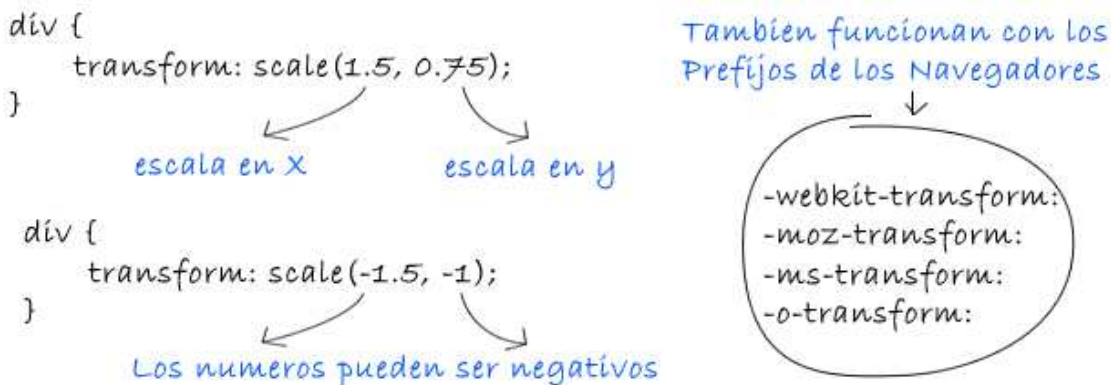
Esta función de la propiedad transform tenemos que aplicarle a transform: translate(x,y); los prefijos de los diferentes navegadores, es decir -ms-, -webkit-, -o- y -ms- para visualizarlo correctamente en los principales navegadores.

Esta función también la podemos usar con valores negativos, es decir (x,y) = (-30px , -10px), y así podemos darle la ubicación deseada.

scale

o Transformación de Escala:

Otra función de la propiedad transform es scale, esta función nos va a permitir escalar un elemento tanto horizontal como verticalmente entorno a su origen y lo hacemos por medio de su (x,y).



Veamos como afecta el siguiente código CSS al h1:

Vamos a mirar el h1 en su estado normal y el h1 con la transformación de escala:



rotate

o Transformación de Rotación:

Otra gran novedad de CSS3, esta función nos va a permitir rotar ya sea un texto, un div u otro elemento del documento HTML, solamente tenemos que indicar los

Propiedades CSS3

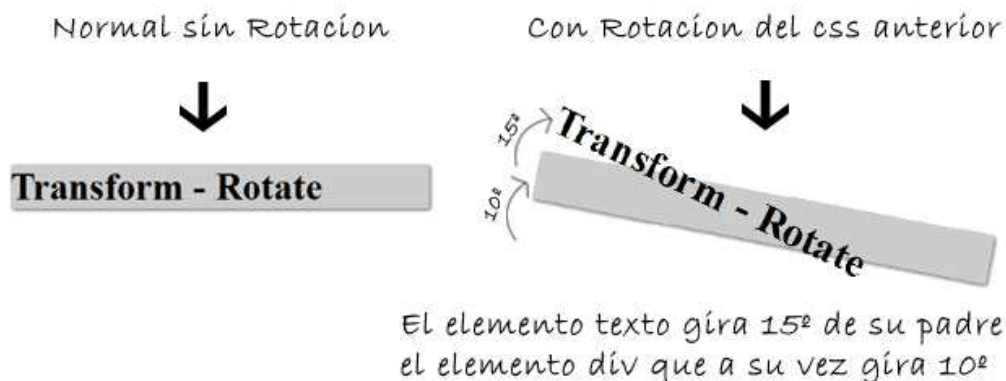
grados que debe girar el elemento ya sea un valor negativo o positivo guiándonos, positivo se moverá como se mueven las manecillas del reloj.



Para el siguiente código CSS:

```
div {
  background: #ccc;
  box-shadow: 2px 2px 3px #777;
  margin: 0 auto;
  transform: rotate(10deg);
  -moz-transform: rotate(10deg);
  -ms-transform: rotate(10deg);
  -o-transform: rotate(10deg);
  -webkit-transform: rotate(10deg);
  width: 350px;
}
h1{
  transform: rotate(15deg);
  -moz-transform: rotate(15deg);
  -ms-transform: rotate(15deg);
  -o-transform: rotate(15deg);
  -webkit-transform: rotate(15deg);
}
```

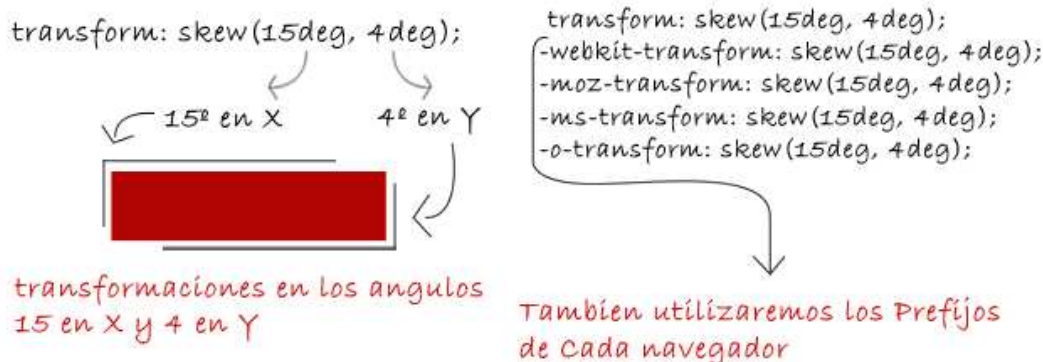
y este código HTML: `<div> <h1>Transform - Rotate</h1> </div>`



skew

o Transformación de Inclinación:

Esta función nos va a permitir inclinar o aplicar un tipo de perspectiva a un elemento en el documento HTML, como los demás tenemos que especificar las coordenadas de sus ángulos para su inclinación, (X, Y) el primero X que se aplica Horizontalmente y Y que se aplica verticalmente.



Para el siguiente código HTML: `<div> <h1>Transform - Skew</h1> </div>`
Vamos a tener el siguiente resultado si se aplican estas reglas CSS:

```
div {  
  background: #ccc;  
  box-shadow: 2px 2px 3px #777;  
  margin: 0 auto;  
  transform: skew(15deg, 4deg);  
  [-webkit-transform: skew(15deg, 4deg);  
  -moz-transform: skew(15deg, 4deg);  
  -ms-transform: skew(15deg, 4deg);  
  -o-transform: skew(15deg, 4deg);]  
  width: 350px;  
}
```

Para una mayor información
ve a esta
página: http://www.w3schools.com/cssref/css3_pr_transform.asp



Transformaciones 3D

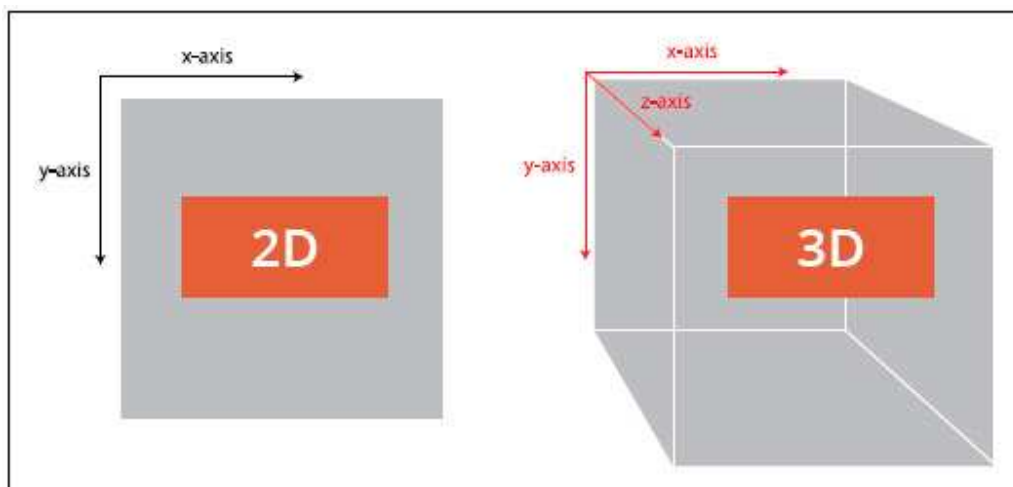
Con CSS3 vamos a poder manipular los elementos en entornos 3D utilizando un nuevo eje, el eje Z, que en adición a los ejes X y Y hacen que el elemento sea tridimensional. Así que en este tutorial vamos a ver los conceptos claves de estas transformaciones 3D que llevan a CSS3 a un plano superior.

¿Qué son las Transformaciones 3D?

Son funciones que nos van a permitir manipular un elemento en un plano tridimensional permitiéndote crear ilusiones de profundidad y distancia

Propiedades CSS3

utilizando los ejes X, Y y Z.



El cuadrado de la izquierda muestra los ejes x e y en un entorno 2D, y el cubo de la derecha incorpora el eje z en un entorno 3D.

Propiedades básicas de Transformaciones 3D en CSS3

CSS3 nos coloca a disposición un grupo de propiedades que vamos a necesitar para trabajar en entornos 3D. Estas son:

- perspective
- perspective-origin
- transform-style
- backface-visibility
- transform (Esta propiedad contiene las funciones que nos permitirán rotar, escalar, mover, sesgar, los elementos tanto en entornos 2D como 3D.)

Así que vamos a ver cada una de ellas.

Perspective

sintaxis

`perspective: longitud|none;`

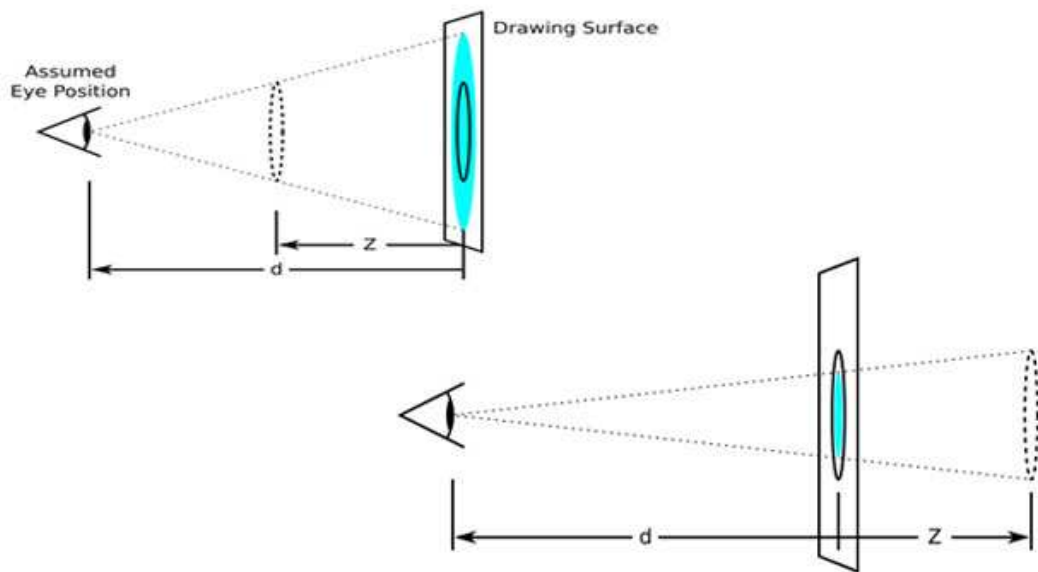
Para crear el contexto 3D y manipular los elementos HTML en ese plano es necesario especificar una perspectiva de visión y añadir la profundidad necesaria, y esto lo conseguimos con la propiedad `perspective`, esta propiedad acepta valores de longitud basada en píxeles que especificamos para establecer la distancia a lo largo del eje Z entre el canvas donde se encuentra el elemento hasta a la vista del espectador.

Cuando definimos la propiedad `perspective` a un elemento solo sus hijos directos recibirán esta vista no el elemento en si.

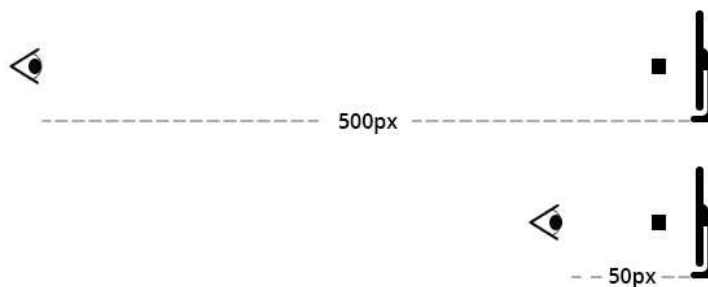
```
.contenedor {  
  perspective: 600px;  
  -webkit-perspective: 600px; /* Chrome, Safari, Opera */  
}
```

Propiedades CSS3

En el código anterior establecemos el entorno 3D por medio de una perspectiva de 600px a lo largo del eje Z.



En el gráfico anterior podemos ver varias cosas, primero el papel que juega el eje Z y segundo que mientras menor sea el valor que indiquemos en la perspectiva mayor profundidad visual tendremos en los elementos y entre mayor valor menor profundidad y mas llana será las modificaciones tridimensionales.



Perspective-origin

sintaxis

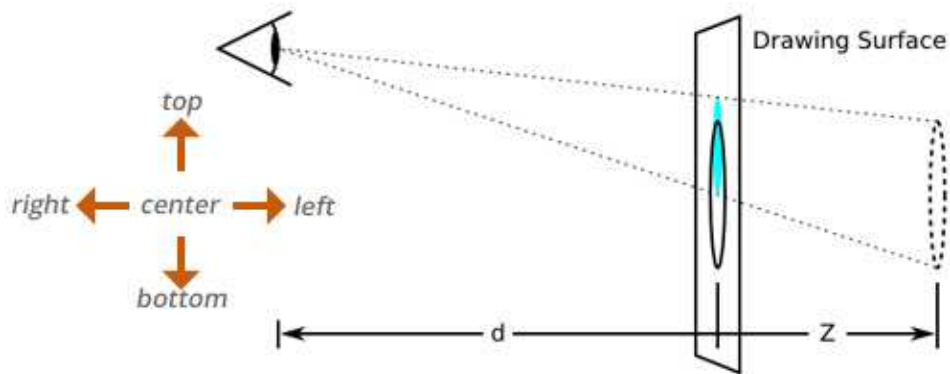
`perspective-origin: eje-x eje-y | valor_inicial | heredado;`

Con esta propiedad vamos a tener un mayor control de la propiedad perspective sobre su entorno 3D.

Define el origen base donde un elemento 3D basa sus ejes X e Y y por ende determina la posición del observador. Su valor inicial es 50% 50% lo que sitúa el punto origen en el centro del elemento.

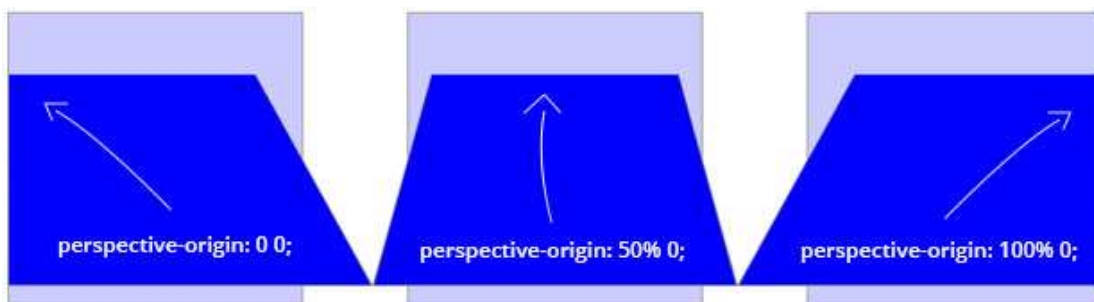
Esta propiedad define dos valores, uno para el eje X y el otro para el eje Y, en ellos podemos colocar valores porcentuales que definirán un punto en el elemento o colocar las palabras claves top, right, bottom, left o center para definir los lados o esquinas del elemento.

Propiedades CSS3



Por ejemplo:

```
.contenedor {  
    perspective-origin: 0 0; /* top left */  
}  
.contenedor {  
    perspective-origin: 50% 0; /* top center */  
}  
.contenedor {  
    perspective-origin: 100% 0; /* top right */  
}
```



Recordemos que en los dos valores que podemos colocar en su argumento vamos a especificar primero el origen X y el segundo valor es para el eje Y.

transform-style

sintaxis

```
transform-style: flat|preserve-3d;
```

Como decíamos al comienzo, con la propiedad `perspective` creamos el entorno 3D y solo los hijos directos del elemento donde aplicamos esta propiedad serán afectados y van a poder ser modificados en este plano. Con la propiedad `transform-style` y su valor `preserve-3d` vamos a poder especificar que este entorno 3D siga afectando a los hijos de los hijos directos del elemento principal.

Propiedades CSS3

```
contenedor > div {  
-webkit-transform-style: preserve-3d; /* Chrome, Safari, Opera */  
transform-style: preserve-3d;  
}
```

Veamos un ejemplo, tenemos este HTML básico con un div principal y dos divs anidados, con sus respectivos estilos:

```
<div class="padre">  
  <div class="hijo">  
    <div class="nieto">  
    </div>  
  </div>  
</div>
```

```
.padre {  
  perspective: 300px;  
}  
hijo {  
  transform: rotateX(45deg);  
}  
.nieto {  
  transform: rotate(20deg); /* No se vera afectado por el entorno 3D */  
}
```

El elemento con clase `.nieto` no se verá afectado por que no es hijo directo del elemento que tiene la perspectiva. Pero si aplicamos la propiedad `transform-style` y como valor `preserve-3d` En el elemento hijo, este preservará el entorno 3D que hereda de su elemento padre para así poder afectar el elemento nieto.

```
.padre {  
  perspective: 300px;  
}  
.hijo {  
  transform: rotateX(45deg);  
  transform-style: preserve-3d;  
}  
.nieto {  
  transform: rotateX(20deg); /* Ahora si tendrá modificaciones 3D */  
}
```

backface-visibility

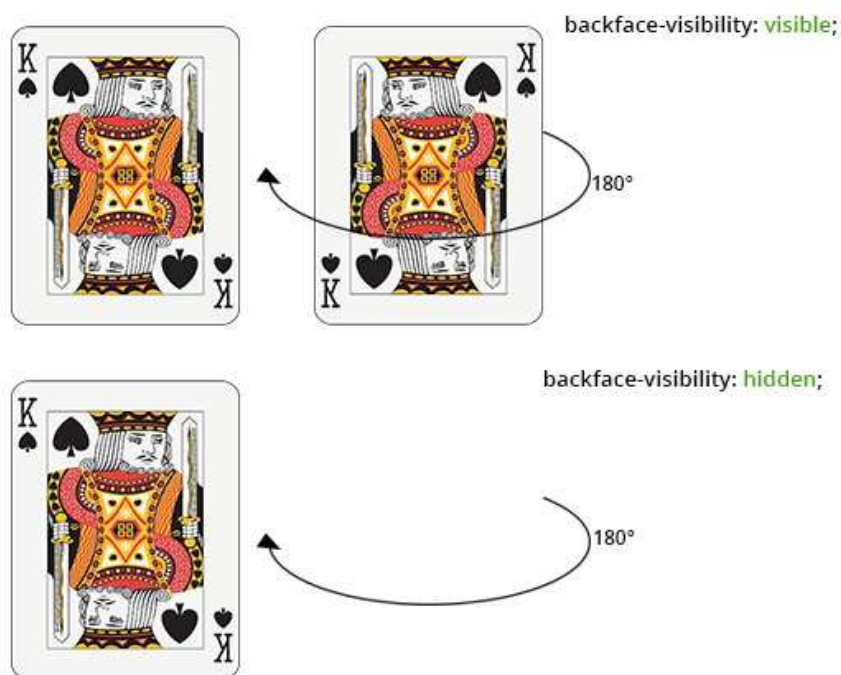
sintaxis

```
backface-visibility: visible|hidden;
```

Cuando un elemento esta en un entorno 3D va a poder ser rotado y la cara posterior va a tener que ser mostrada, por ejemplo cuando hace un giro de 180°. La propiedad `backface-visibility` nos va a permitir especificar si la cara o lado

Propiedades CSS3

posterior de un elemento va a ser mostrado cuando este sea girado, sus valores son `hidden` y `visible` (que es valor por defecto) y hace que se muestre la parte posterior del elemento.



Y ahí es donde aprovechamos para mostrar otro elemento con un lado con una imagen o contenido diferente, lo que conocemos como el efecto “flip” en CSS3. [Ver aquí su resultado http://codepen.io/tutosytips/pen/kfjal.](http://codepen.io/tutosytips/pen/kfjal)

Box-Sizing

La propiedad `box-sizing` se usa para modificar las propiedades por defecto del [CSS box model](#) que calculan el alto y el ancho de los elementos . De hecho es posible usarla para emular el comportamiento de los navegadores que no soportan las especificaciones del [CSS box model](#).

Se aplica a todos a todos los elementos que acepten la propiedad `width` no siendo heredable. Puede adoptar diferentes valores:

content-box

Este es el estilo por defecto especificado en el estándar de CSS. Las propiedades `width` y `height` son las medidas incluidas solo en el contenido, pero no incluyen el padding, el border o el margin. Por ejemplo, el elemento `.box {width: 300px; border: 10px solid black; padding: 10px; }` cuando sea interpretado por el navegador tendrá un `width` de 340px (resultado de la suma del border, el padding y el width).

Propiedades CSS3

padding-box

En este caso las propiedades [width](#) y [height](#) incluyen en padding pero no incluyen el border y el margin.

border-box

El [width](#) y el [height](#) incluyen el padding y el border, pero no el margin. Volviendo al primer ejemplo, si usamos border-box en el elemento .box {width: 300px; border: 10px solid black; padding: 10px; -moz-box-sizing: border-box; box-sizing: border-box; } tendrá un width de 300px después de ser interpretado por el navegador.

Como ejemplo, teniendo los siguientes estilos que se aplican sobre estos elementos html:

<pre>.div1 { width: 300px; height: 100px; border: 1px solid blue; } .div2 { width: 300px; height: 100px; padding: 50px; border: 1px solid red; } .div3 { width: 300px; height: 100px; border: 1px solid blue; box-sizing: border-box; } .div4 { width: 300px; height: 100px; padding: 50px; border: 1px solid red; box-sizing: border-box; }</pre>	<pre><h2>Sin aplicar box-sizing</h2> <div class="div1">Este div es pequeño (con width de 300px y height de 100px).</div>
 <div class="div2">Este div es grande (aunque sus propiedades width y height sean iguales).</div> <h2>Aplicando box-sizing</h2> <div class="div3">Los dos divs son iguales ahora</div>
 <div class="div4">Perfecto!</div></pre>
--	---

Obtendremos:

Sin aplicar box-sizing

Este div es pequeño (con width de 300px y height de 100px).

Este div es grande (aunque sus propiedades width y height sean iguales).

Aplicando box-sizing

Los dos divs son iguales ahora

Perfecto!

Cajas flexibles

Las cajas flexibles, o flexbox, es un nuevo modo de disposición en CSS3.

El uso de flexbox asegura que los elementos se comporten de manera predecible cuando el diseño de la página debe acomodar diferentes tamaños de pantalla y diferentes dispositivos de visualización.

Para muchas aplicaciones, el modelo de caja flexible proporciona una mejora sobre el modelo de bloque en el sentido de que no utiliza floats, ni tampoco los márgenes del contenedor flexible se colapsan con los márgenes de su contenido.

Flexbox se compone de contenedores y elementos flexibles.

Se declara un contenedor flexible estableciendo la propiedad de visualización de un elemento a tipo flex (renderizándose como un bloque) o a `inline-flex` (representándose como elemento en línea).

Propiedades CSS3

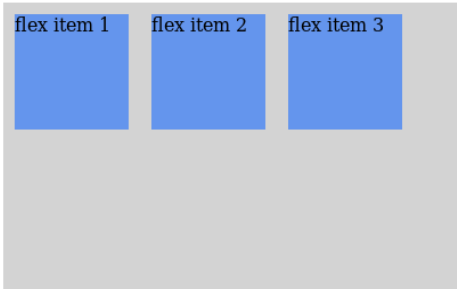
Dentro de un contenedor flexible hay uno o más elementos flexibles.

Nota: todo lo que se encuentre fuera de un contenedor flexible y dentro de un elemento flexible se procesa como de costumbre. Flexbox define cómo se colocan los elementos flexibles dentro de un contenedor flexible.

Los elementos flexibles se colocan dentro de un contenedor flexible a lo largo de una línea flexible. Por defecto, sólo hay una línea flexible por contenedor flexible.

Display:flex

El siguiente ejemplo muestra tres elementos flexibles. Están posicionados de forma predeterminada: a lo largo de la línea horizontal de flexión, de izquierda a derecha:

<pre>.flex-container { display: -webkit-flex; display: flex; width: 400px; height: 250px; background-color: lightgrey; } .flex-item { background-color: cornflowerblue; width: 100px; height: 100px; margin: 10px; }</pre>	<pre><div class="flex-container"> <div class="flex-item">flex item 1</div> <div class="flex-item">flex item 2</div> <div class="flex-item">flex item 3</div> </div></pre> 
---	--

flex-direction

También es posible cambiar la dirección de la línea de flexión.

Si ajustamos la propiedad de direction a rtl (de derecha a izquierda), el texto se dibuja de derecha a izquierda y también la línea de flexión cambia de dirección, lo que cambiará el diseño de página.

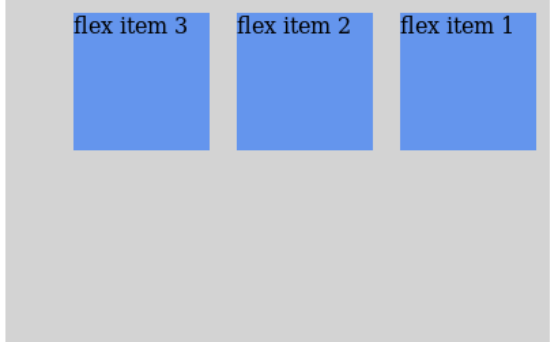
La propiedad de direction de flex especifica la dirección de los elementos flexibles dentro del contenedor de flexión. El valor predeterminado de la dirección de flexión es fila (de izquierda a derecha, de arriba a abajo).

Los otros valores son los siguientes:

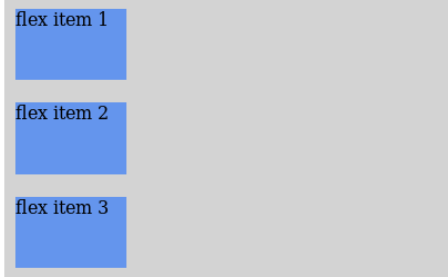
Propiedades CSS3

- row-reverse - Si el modo de escritura (dirección) se deja a la derecha, los elementos de flexión se colocarán de derecha a izquierda
- column - Si el sistema de escritura es horizontal, los elementos de flexión serán dispuestos verticalmente
- column-reverse - Igual que la columna, pero invertida

El siguiente ejemplo muestra el resultado del uso del valor de fila inversa:

<pre>.flex-container { display: -webkit-flex; display: flex; -webkit-flex-direction: row-reverse; flex-direction: row-reverse; width: 400px; height: 250px; background-color: lightgrey; } .flex-item { background-color: cornflowerblue; width: 100px; height: 100px; margin: 10px; }</pre>	<pre><div class="flex-container"> <div class="flex-item">flex item 1</div> <div class="flex-item">flex item2</div> <div class="flex-item">flex item 3</div> </div></pre> 
---	--

El siguiente ejemplo muestra el resultado de usar el valor column:

<pre>.flex-container { display: -webkit-flex; display: flex; -webkit-flex-direction: column; flex-direction: column; width: 400px; height: 250px; background-color: lightgrey; } .flex-item { background-color: cornflowerblue; width: 100px; height: 100px; margin: 10px; }</pre>	<pre><div class="flex-container"> <div class="flex-item">flex item 1</div> <div class="flex-item">flex item 2</div> <div class="flex-item">flex item 3</div> </div></pre> 
---	--

Si usamos flex-direction: column-reverse; los elementos se dispondrían empezando por el tercero.

Justify-content

La propiedad Justify-content alinea horizontalmente los elementos del contenedor flexible cuando los elementos no utilizan todo el espacio disponible en el eje principal.

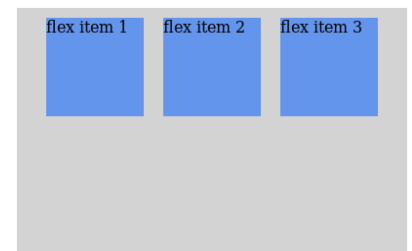
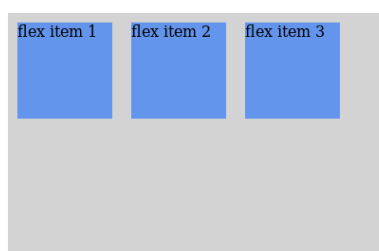
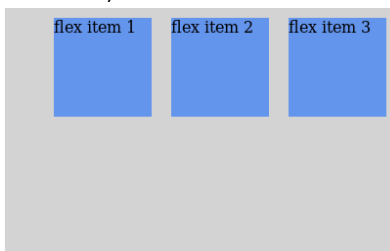
Los valores posibles son los siguientes:

- Flex-start - Valor predeterminado. Los artículos se colocan al principio del contenedor
- Flex-end - Los artículos se colocan en el extremo del contenedor
- Center - Los objetos se colocan en el centro del contenedor
- Space-between - Los elementos se colocan con espacio entre las líneas
- Space-around - Los elementos se colocan con espacio antes, entre y después de las líneas

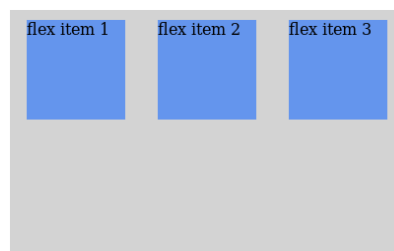
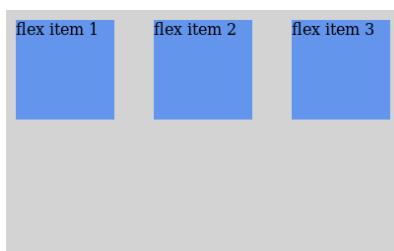
Ejemplos al utilizar los diferentes valores sobre estos elementos con estas reglas:

<pre>.flex-container { display: -webkit-flex; display: flex; width: 400px; height: 250px; background-color: lightgrey; } .flex-item { background-color: cornflowerblue; width: 100px; margin: 10px; }</pre>	<pre><div class="flex-container"> <div class="flex-item">flex item 1</div> <div class="flex-item">flex item 2</div> <div class="flex-item">flex item 3</div> </div></pre>
---	---

justify-content: flex-end; justify-content: flex-start; justify-content: center;



justify-content: space-between; justify-content: space-around;



Propiedades CSS3

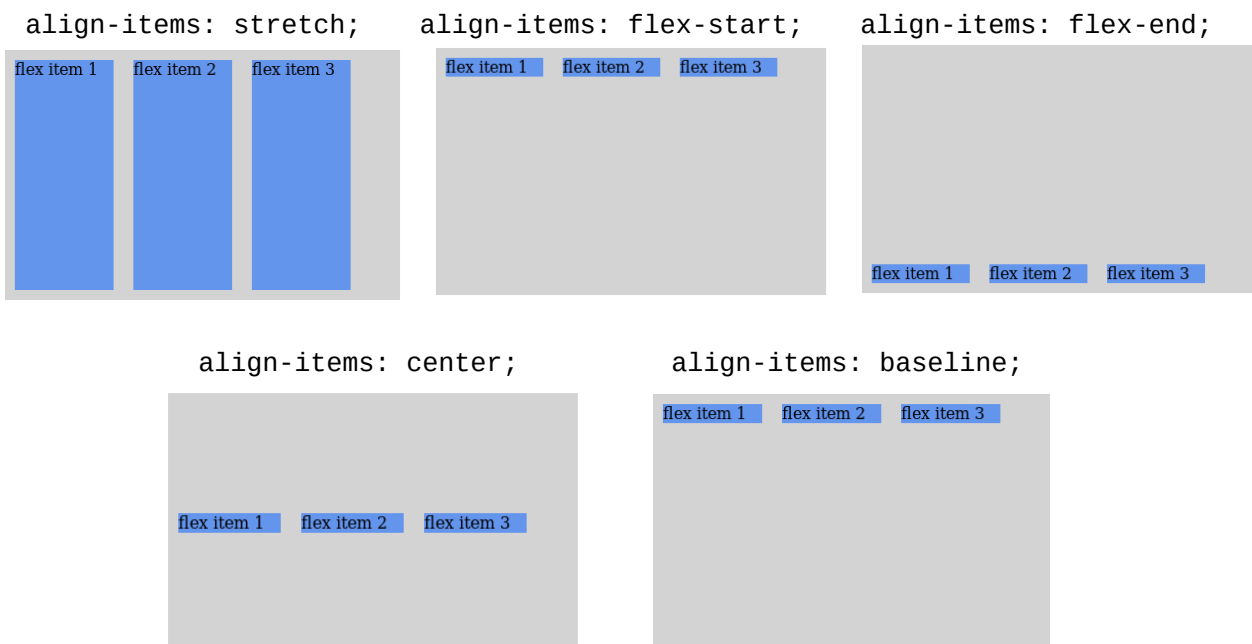
align-items

La propiedad align-items alinea verticalmente los elementos del contenedor flexible cuando los elementos no utilizan todo el espacio disponible en el eje transversal.

Los valores posibles son los siguientes:

- stretch - Valor predeterminado. Los artículos se estiran para ajustar el contenedor
- flex-start - Los elementos se colocan en la parte superior del contenedor
- flex-end - Los artículos se colocan en la parte inferior del contenedor
- center - Los artículos se colocan en el centro del contenedor (verticalmente)
- baseline - Los elementos se colocan en la línea de base del contenedor

Ejemplos al utilizar los diferentes valores sobre los elementos del ejemplo anterior:



flex-wrap

La propiedad flex-wrap especifica si los elementos flexibles deben envolverse o no, si no hay suficiente espacio para ellos en una línea flex.

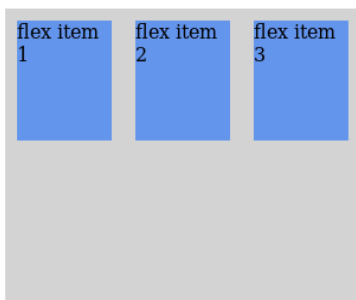
Propiedades CSS3

Los valores posibles son los siguientes:

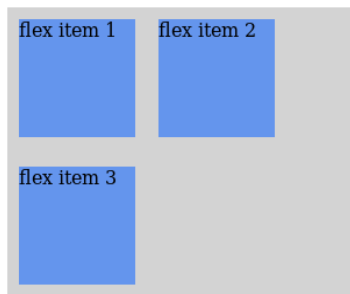
- Nowrap - Valor predeterminado. Los elementos flexibles no se envolverán
- Wrap - Los artículos flexibles se envolverán si es necesario
- Wrap-reverse - Los artículos flexibles se envolverán, si es necesario, en orden inverso

Ejemplos al utilizar los diferentes valores sobre los elementos del ejemplo anterior aplicando también un height de 100px a .flex-item:

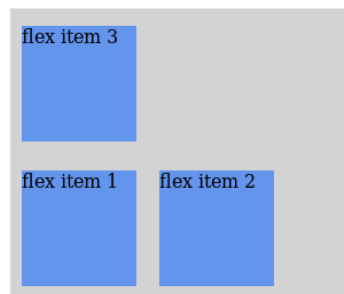
`flex-wrap: nowrap;`



`flex-wrap: wrap;`



`flex-wrap: wrap-reverse;`



align-content

La propiedad align-content modifica el comportamiento de la propiedad flex-wrap. Es similar a align-items, pero en lugar de alinear elementos flexibles, alinea las líneas flexibles.

Los valores posibles son los siguientes:

- stretch - Valor predeterminado. Las líneas se extienden para ocupar el espacio restante
- flex-start - Las líneas están empacadas hacia el inicio del contenedor flexible
- flex-end - Las líneas se empaquetan hacia el extremo del contenedor flexible
- center - Las líneas están empaquetadas hacia el centro del contenedor flexible
- space-between - Las líneas están uniformemente distribuidas en el contenedor flexible
- space-around - Las líneas están distribuidas uniformemente en el contenedor flexible, con espacios de tamaño medio en cada extremo

El siguiente ejemplo muestra el resultado de usar el valor del centro:



Propiedades CSS3

Propiedades de los elementos flexibles.

Orden

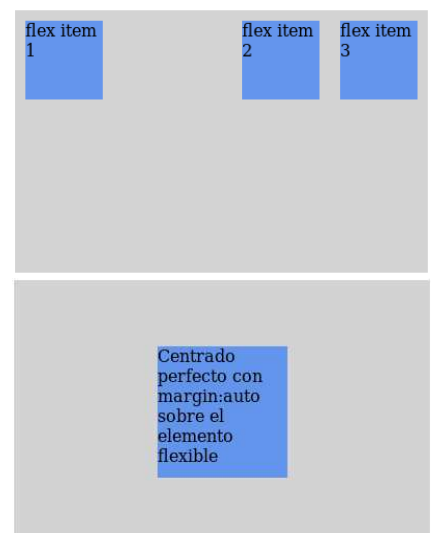
La propiedad `order` especifica el orden de un elemento flexible en relación con el resto de los elementos flexibles dentro del mismo contenedor:

<pre>.flex-container { display: -webkit-flex; display: flex; width: 400px; height: 250px; background-color: lightgrey; } .flex-item { background-color: cornflowerblue; width: 100px; height: 100px; margin: 10px; } .first { -webkit-order: -1; order: -1; }</pre>	<pre><div class="flex-container"> <div class="flex-item">flex item 1</div> <div class="flex-item first">flex item 2</div> <div class="flex-item">flex item 3</div> </div></pre> 
---	---

Margen

Estableciendo `margin: auto;` absorberá el espacio extra. Se puede utilizar para empujar elementos flexibles en diferentes posiciones.

En el siguiente ejemplo establecemos `margin-right: auto;` para el primer elemento flex. Esto hará que todo el espacio extra sea absorbido a la derecha de ese elemento:



Centrado perfecto

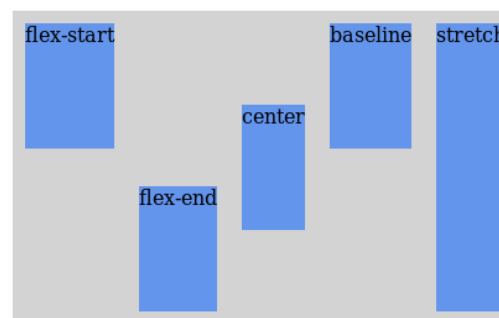
En el siguiente ejemplo resolveremos un problema casi diario: centrado perfecto.

Es muy fácil con flexbox. `margin: auto;` hará que el elemento quede perfectamente centrado en ambos ejes:

Alineado

La propiedad `align-self` de los elementos flexibles anula la propiedad `align-items` del contenedor flex para ese elemento. Tiene los mismos valores posibles que la propiedad `align-items`.

El siguiente ejemplo establece diferentes valores de `align-self` para cada elemento flexible:



Propiedad flex

La propiedad `flex` especifica la longitud del elemento flex, en relación con el resto de los elementos flexibles dentro del mismo contenedor.

En el siguiente ejemplo, el primer elemento flexible consumirá 2/4 del espacio libre y los otros dos elementos flexibles consumirán 1/4 del espacio libre cada uno:

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}  
  
.flex-item {  
  background-color: cornflowerblue;  
  margin: 10px;  
}  
  
.item1 {-webkit-flex: 2;flex: 2;}  
.item2 {-webkit-flex: 1;flex: 1;}  
.item3 {-webkit-flex: 1;flex: 1;}
```

```
<div class="flex-container">  
  <div class="flex-item item1">flex  
  item 1</div>  
  <div class="flex-item item2">flex  
  item 2</div>  
  <div class="flex-item item3">flex  
  item 3</div>  
</div>
```



Media queries (consultas de medios)

La regla `@media`, introducida en CSS2, permitió definir diferentes reglas de estilo para diferentes tipos de medios.

Ejemplos: Puede tener un conjunto de reglas de estilo para pantallas de computadora, una para impresoras, una para dispositivos portátiles, otra para dispositivos de tipo televisión, etc.

Propiedades CSS3

Desafortunadamente estos tipos de medios nunca recibieron mucha compatibilidad con dispositivos, aparte del tipo de medios de impresión.

Las consultas de medios en CSS3 amplían la idea de los tipos de medios CSS2: En lugar de buscar un tipo de dispositivo, buscan la capacidad del dispositivo.

Las consultas de medios se pueden utilizar para comprobar muchas cosas, tales como:

- Anchura y altura de la ventana del dispositivo
- Anchura y altura del dispositivo
- Orientación (esta tableta / teléfono en apaisado o vertical?)
- Resolución de pantalla

El uso de consultas de medios es una técnica popular para entregar una hoja de estilos adaptada a tabletas y teléfonos.

Sintaxis

Una media query consiste en un tipo de medio que puede contener una o más expresiones, que se resuelven a true o false.

```
@media not|only tipo_de_medio and (expresiones) {  
    Código CSS;  
}
```

El resultado de la consulta es cierto si el tipo de medio especificado coincide con el tipo de dispositivo en el que se muestra el documento. Cuando una consulta de medios es verdadera, se aplican las reglas de estilo o de hoja de estilo correspondientes, siguiendo las reglas en cascada habituales.

A menos que utilice los operadores not o only, el tipo de medio es opcional y el tipo all se usará por defecto.

También puede tener diferentes hojas de estilo para diferentes soportes:

```
<link rel="stylesheet" media="tipo_de_medio and|not|only (expresiones)"  
href="para_impresoras.css">
```

Ejemplos:

Una forma de usar consultas de medios es tener una sección CSS alternativa dentro de su hoja de estilo.

El siguiente ejemplo cambia el color de fondo a lightgreen si la ventana es de 480 píxeles o más de ancho (si la vista es menor de 480 píxeles, el color de fondo será de color rosa):

Propiedades CSS3

```
@media screen and (min-width: 480px) {  
  body {  
    background-color: lightgreen;  
  }  
}
```

Cuando el ancho del navegador es entre 520 y 699px, se aplicará un icono de correo electrónico a cada enlace de correo electrónico:

```
@media screen and (max-width: 699px) and (min-width: 520px) {  
  ul li a {  
    padding-left: 30px;  
    background: url(email-icon.png) left center no-repeat;  
  }  
}
```

Para anchos de navegador por encima de 1151px, volveremos a agregar el icono como lo usamos antes.

Aquí, no tenemos que escribir un bloque adicional de consulta de medios, solo podemos agregar una consulta de medios adicional a nuestra ya existente usando una coma (esto se comportará como un operador OR):

```
@media screen and (max-width: 699px) and (min-width: 520px), (min-width: 1151px)  
{  
  ul li a {  
    padding-left: 30px;  
    background: url(email-icon.png) left center no-repeat;  
  }  
}
```

Selectores:

CSS3 introduce algunos selectores avanzados:

- E:only-of-type: un elemento que es único en su tipo.
- E:not(s): un elemento que no coincide con los selector simples.
- E ~ F: un elemento F precedido de un elemento E.
- E:nth-child: un objeto que es el enésimo hijo del nodo padre.
- E:nth-last-child: un objeto que es el último hijo del nodo padre.
- E:nth-of-type: un elemento que es el enésimo nodo hijo de otro elemento.
- E.first-of-type: el primer elemento de un tipo.

Propiedades CSS3

Este documento ha sido elaborado basándome en:

Documentación oficial del estándar CSS3 de la W3C:

http://www.w3schools.com/css/css3_intro.asp

Diferentes tutoriales extraídos de la página:

<http://www.tutosytips.com/>

Imobach Ramírez Fos, 2016:

Web ---> <http://imobach.pro>

Blog ---> <https://systemctl.ml>

