IT614

More CSS3 Properties

CSS3 is the latest evolution of the Cascading Style Sheets language and it brings a lot of long awaited novelties, like rounded corners, shadows, gradients, transitions or animations, as well as new layouts like multi-columns , flexible box or grid layouts. The leading browsers are still in catch-up mode. Not all the properties supported by current leading browsers. When a browser does not implement the standard, you need to vendor specific prefixes: -webkit- for Chrome and Safari, -ms- for IE, -moz- for Firefox, and –o- for Opera.

If you want to develop browser independent We page, you need to include all of them (including no prefix). In my notes whenever needed, I use –webkit- (because I use Chrome).

Throughout these notes, I have referred to these Web sites:

http://www.w3.org/TR/css3-animations/
https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_animations
and
http://msdn.microsoft.com/en-us/library/ie/jj680076(v=vs.85).aspx

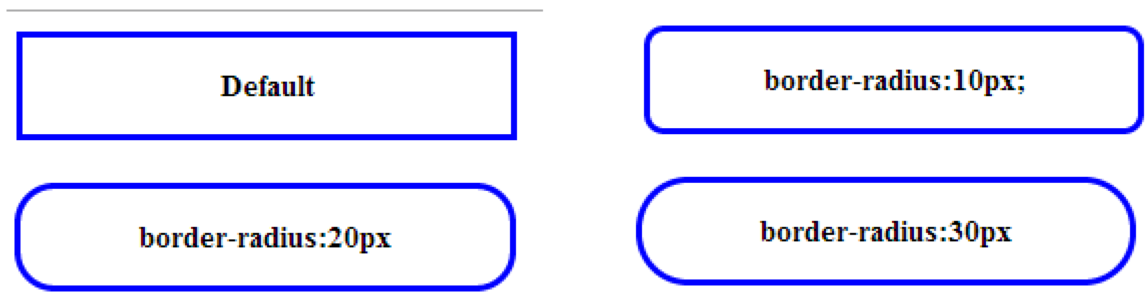
There are plenty of Web sites, which detailed browser support for CSS3 properties.

Some of them:
http://caniuse.com/
http://css3test.com/
http://www.w3schools.com/cssref/css3_browsersupport.asp
http://html5test.com/

1. CSS property: border-radius

As you know, the display of the content of an element is enclosed in a box. By default the corners of this box are rectangles (not rounded). The **border-radius** property makes the corners round. The value is in usual length units, representing the radius of the corner. Instead of explaining with circles with different radii in the corners, let me show some examples to illustrate how different value effect the corners.

Study the output for different values of border-radius property:



The corners are arcs of circles of specified radii.

The `border-radius` CSS property sets the border-corners to be rounded in all four corners. There are five variations of **border-radius** property:

border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius and **border-radius.**

The first four properties **border-top-left-radius, border-top-right-radius, border-bottom-right-radius,** and **border-bottom-left-radius** control the border corners, as indicated by the name. The property **border-radius**, controls all the four corners.

The values of these properties are length in usual units.

Property	**Meaning**
border-top-left-radius : *value*	Sets the top left corner to radius value
border-top-right-radius: *value*	Sets the top right corner to radius value
border-bottom-right-radius: *value*	Sets the bottom right corner to radius value
border-bottom-left-radius: *value*	Sets the bottom left corner to radius value
border-radius: *value*	Sets the all four border thickness to radius *value*

The property **border-radius** can take 1 to 4 values:

- **border-radius with one value**

border-radius: *value;*
Sets all the four corners to the radius *value*.

Example:
border-radius: 18px;

Sets all the four corners to the radius 18 pixels.

- **border-radius with two values**

border-radius: *value1 value2;*
Sets top-left and bottom-right corner radii to *value1* and top-right and bottom-left corner radii to *value2*.

Example:
border-radius: 15px thick;

Sets top-left and bottom-right corner radii to 15 pixels and top-right and bottom-left corner radii to **thick**.

- **border-radius with three values**

border-radius: *value1 value2 value3;*

Sets *top-left* corner radius to *value1*, *top-right and bottom-left* corner radii to *value2*, and *bottom-right* corner radius to *value3*.

Example:
border-radius: 10px 15px 10px;

Sets *top-left* corner radius to *10px*, *top-right and bottom-left* corner radii to *15px*, and *bottom-right* corner radius to *10px*.

- **border-radius with four values**

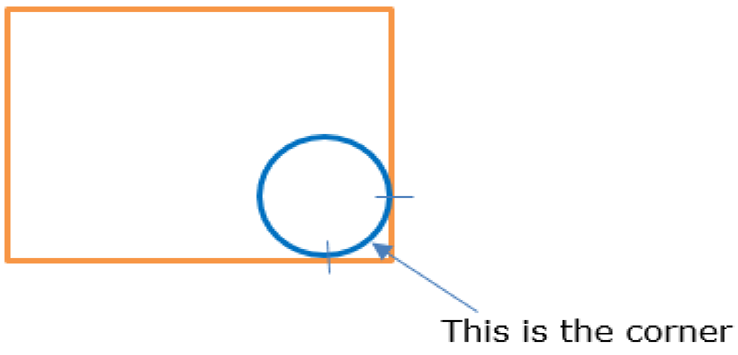
border-radius: *value1 value2 value 3 value4;*

Sets *top-left*, *top-right*, *bottom-right*, and *bottom-left* corner radii to *value1, value2, value 3,* and *value4* respectively.

Example:
border-radius: *10px 15px 10px 15px;*

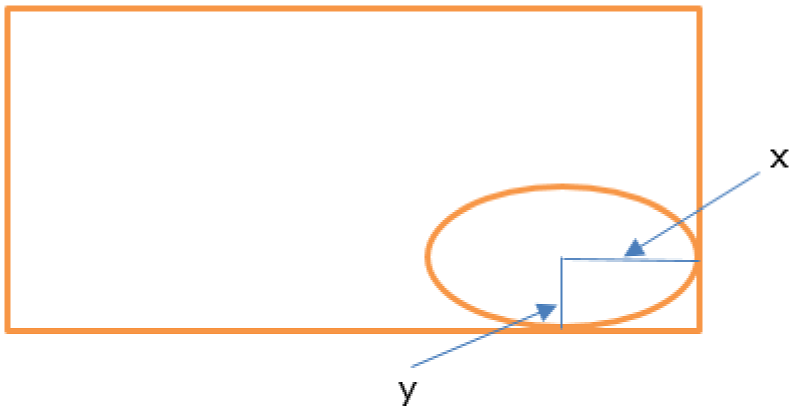
Sets *top-left*, *top-right*, *bottom-right*, and *bottom-left* corner radii to *10px, 15px, 10px, and 15px* respectively.

Arcs of ellipses as corners
As mentioned, border-radius property makes the corners of the display box rounded. The corners are actually arcs of a circle:



The corner arc will be symmetric.

Instead of arc of a circle, you can make a corner an arc of an ellipse:



The x and y values determine the shape of the corner.

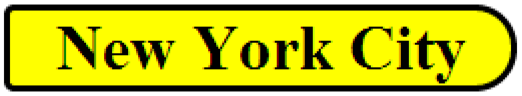
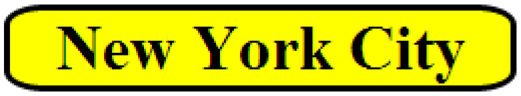
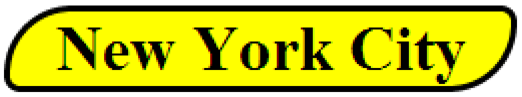
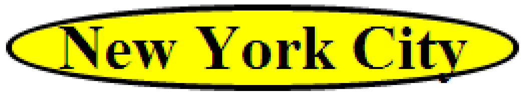
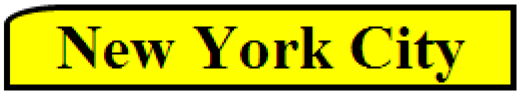
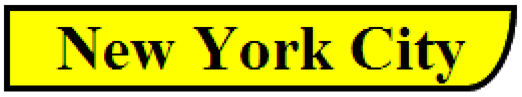
Syntax for elliptic corners:
Each of these, **border-top-left-radius, border-top-right-radius, border-bottom-right-radius,** and **border-bottom-left-radius** takes two values, corresponding to x and y values.

For **border-radius** the x values and y-values are separated by /.
You can go crazy checking all possible combinations.

EXAMPLES:
In all our examples, we will use,

```
<style>
h1{
   background: yellow;
   border-style: solid;
   width:250px;
   text-align:center;
/* We will insert here various border-radius properties. See examples below */
}
```

```
</style>
<body>
  <h1>New York City</h1>
</body>
```

border-radius declaration	**Output**
border-radius: 10px 40px 40px 10px;	
border-radius: 20px/10px;	
border-radius: 40px 10px;	
border-radius: 50%;	
border-top-left-radius: 30px 10px;	
border-bottom-right-radius: 30px 50px;	

NOTE:
Since rounded corner elements and border-radius were not a set standard, each browser implemented their own prefixed {prefix}-border-radius implementation. Those prefixes look like:
-moz-border-radius: 20px;
-webkit-border-radius: 20px;
If your browser works without these prefixes, don't use them.

2. The box-shadow property

The **box-shadow** CSS property describes one or more shadow effects as a comma-separated list.

General syntax:
The **box-shadow** property takes up to 5 values:
box-shadow : *value1 value2 value3 value4 value5*

value1:
This specifies the horizontal offset of the shadow, positive means the shadow will be on the right of the box, a negative offset will put the shadow on the left of the box.

value2:
This specifies the vertical offset of the shadow, a negative one means the box-shadow will be above the box, a positive one means the shadow will be below the box.

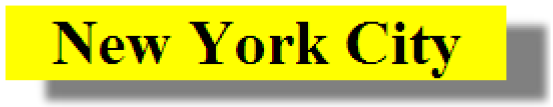
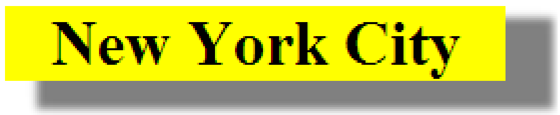
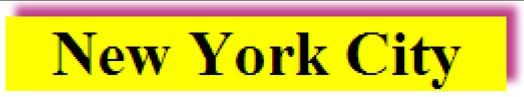
value3:
This specifies the blur radius (optional), if set to 0 the shadow will be sharp, the higher the number, the more blurred it will be.

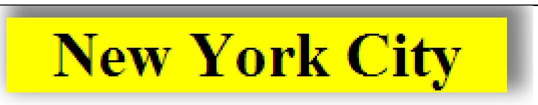
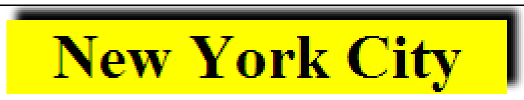
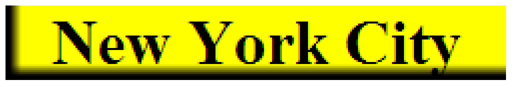
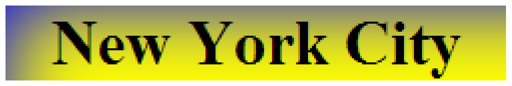
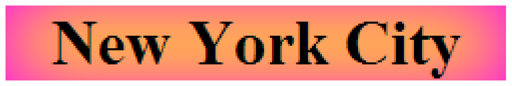
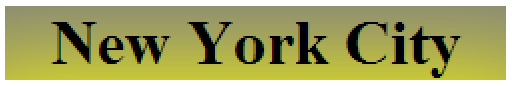
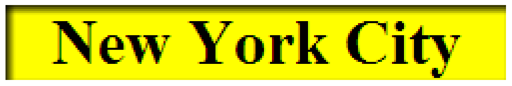
value4:
This specifies the spread radius (optional), positive values increase the size of the shadow, negative values decrease the size. Default is 0 (the shadow is same size as blur).

value5:
This specifies the color of the shadow.

NOTE:
In addition to having up to five values, you can provide another value **inset**. If not specified (default), the shadow is assumed to be a drop shadow (as if the box were raised above the content). The presence of the inset keyword changes the shadow to one inside the frame (as if the content was depressed inside the box). Inset shadows are drawn inside the border (even transparent ones), above the background, but below content.

EXAMPLES:
```
h1{
   background: yellow;
   width:250px;
   text-align:center;
   /* We will insert here box-shadow property with various values. See examples below */
}
  </style>
  <body>
     <h1>New York City</h1>
  </body>
```

box-shadow : *values*	**Result**
box-shadow: 20px 10px 5px 0px gray;	
box-shadow: 20px 10px 5px 4px gray;	
box-shadow: 5px -5px 5px 0px #BF418C;	

box-shadow: 5px -5px 14px 4px gray;	
box-shadow: 5px -5px 5px 0px black;	
box-shadow: inset 5px -5px 5px 0px black;	
box-shadow : inset 1em 1em 2em -1em blue;	
box-shadow : inset 0 0 2em magenta;	
box-shadow: inset 0 2em 3em -1em gray;	
box-shadow: inset 2px 2px 5px black;	

3. Linear Gradients

A gradient is an image that smoothly fades from one color to another. These are commonly used for subtle shading in background images, buttons, and many other things.

As per W3C, a linear gradient is created by specifying a gradient-line and then several colors placed along that line. The image is constructed by creating an infinite canvas and painting it with lines perpendicular to the gradient-line, with the color of the painted line being the color of the gradient-line where the two intersect. This produces a smooth fade from each color to the next, progressing in the specified direction. (http://dev.w3.org/csswg/css-images-3/)

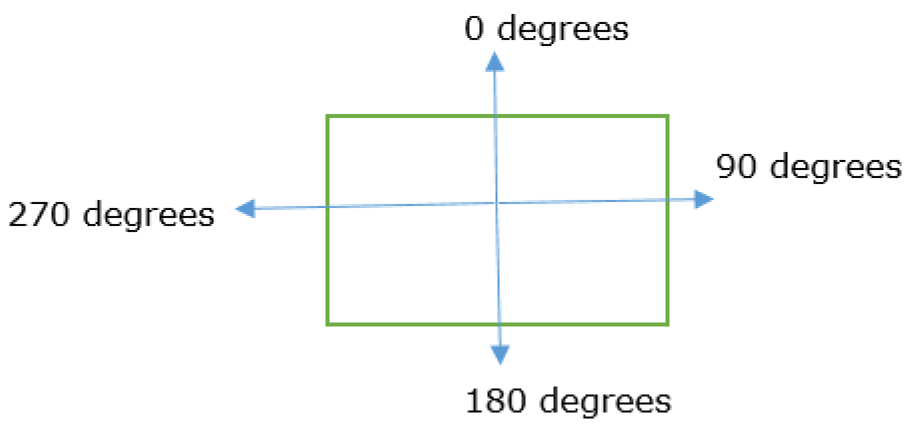
Linear-gradient() uses function notation. And is assigned to the CSS property **background**.

linear-gradient (*argument1, argument2*);

argument1: There are two ways of specifying argument1:

1. An angle: An angle of direction for the gradient. You can specify using degrees (deg) or radians (rad). Examples: 45deg, 90deg, 0rad, 0.5rad.
2. Using key words: **to top, to bottom, to left** and **to right**.
   These are equivalent to the angles 0deg, 180deg, 270deg, 90deg respectively.

The angles are represented as shown below (the rectangle is the CSS element box):

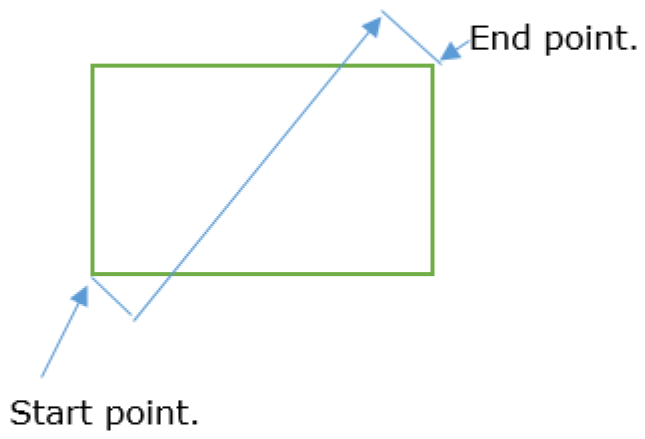


Argument 1	**Starting point**	**Ending point**
0deg (same as **to top**)	bottom edge	top edge
90deg (same as **to right**)	left edge	right edge
180deg (same as **to bottom**)	top edge	bottom edge
270deg (same as **to left**)	Right edge	left edge

NOTE:
If argument1 is omitted, then it defaults to "**to bottom**".

NOTE:
Starting point and the end point are calculated similarly for other angles.

Example:
For angle 60 degrees, draw line at 60^0 at the center of the rectangle:



argument2: There are two ways of specifying argument2:

1. A set of color values, called color stops, separated by commas. The list of colors specify that the colors have to be placed evenly along the gradient line.

2. Each color in the list may also include a percentage number (or a decimal number). The percentage/decimal number indicates where along the gradient ray to place the color stop. "0%" indicates the start of the gradient ray, and "100%" indicates the point where the gradient ray intersects the ending shape. For instance, a value of "20%" indicates the color stop should be placed at a point 20% of the length of the gradient ray, starting from the beginning of the line. Make sure the first color is 0% (that is the default).

Let us take several examples:

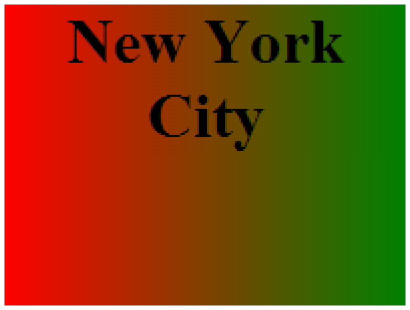
In our examples we use the following:

```
<style>
h1{
   width:200px;
   height:150px;
   text-align:center;
/* We will insert here background property with linear-gradient() with different arguments.*/
}
</style>
<body>
  <h1>New York City</h1>
</body>
```

EXAMPLE:

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

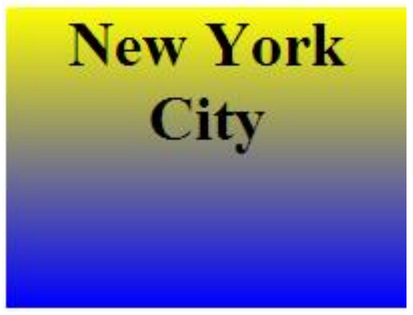
Color red starts on the left edge and ends in green on the right edge:



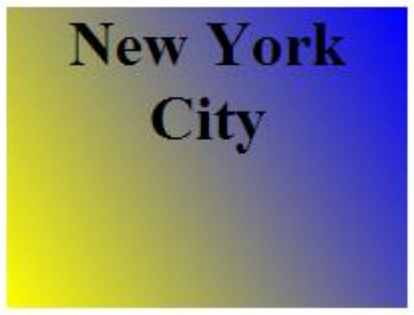
EXAMPLE:

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

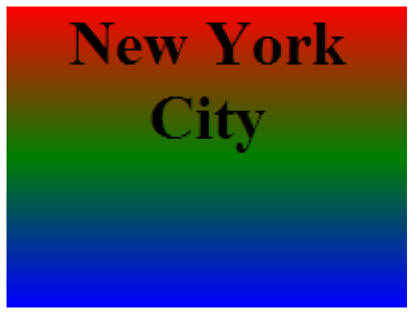
Color yellow starts on the top edge and ends in blue on the bottom edge:



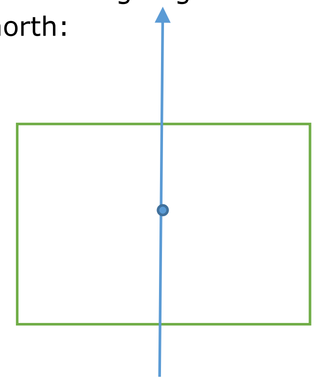
Example:
background: linear-gradient(60deg, yellow, blue);
Color yellow starts at the starting point and ends in blue. (You have to calculate the start and end point using the method described earlier.)



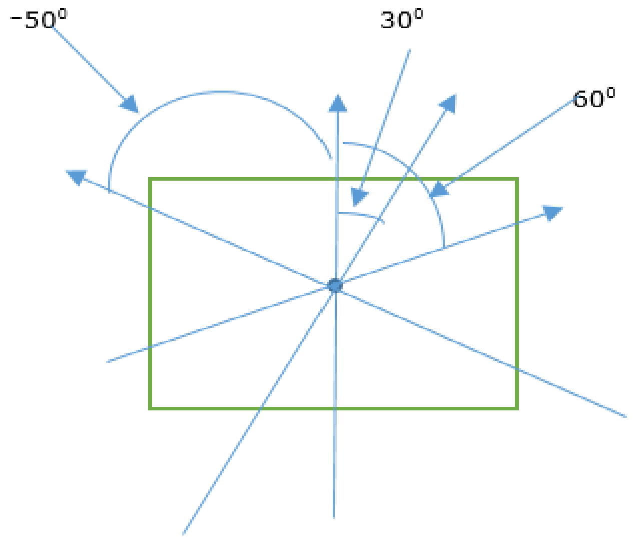
Example:
background: linear-gradient(to bottom, red, green, blue);



One more review on measuring angles: First draw a long arrow line through the center of the rectangle facing north:



To mark a given angle, turn the arrow line through that angle. The following diagram shows some examples:



The arrow head always points to the end point.
Notice that the angle -50^0 is equivalent to 310^0, 60^0 is equivalent to -300^0 and so on. That is, you can arrive at an angle in either clockwise direction (considered +) or in counterclockwise direction (considered -).

EXAMPLE:
All the following are equivalent:

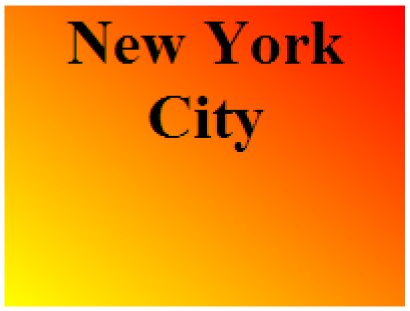
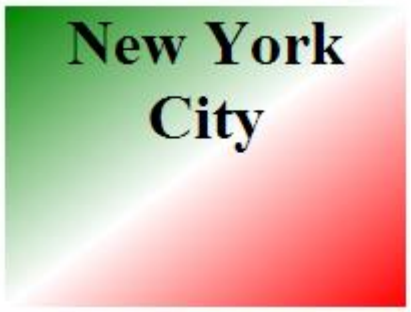
```
linear-gradient(yellow, blue);
linear-gradient(to bottom, yellow, blue);
linear-gradient(180deg, yellow, blue);
linear-gradient(to top, blue, yellow);
```

EXAMPLE:
The following two are equivalent:

```
linear-gradient(70deg, yellow, red);
linear-gradient(-110deg, red,yellow);
```
If this is not clear, draw a diagram.

NOTE:
In addition to using to top, to bottom, to left and to right for the end points, you can also use, **to top left, to top right, to bottom right** and **to bottom left.** Meanings of these are self-explanatory.

Example:

background: linear-gradient(to top right, green, red);	
background: linear-gradient(to bottom right, green, white, red);	

Note:
Earlier versions of linear-gradient() used keywords like top, left, right and bottom. They are obsolete now. Also, when browsers did not provide support for the standard usage, we had to use browser specific prefixes such as –webkit-, and –moz-. The new browsers provide full support now. So these prefixes are not needed any more.

4. Radial gradients

(http://dev.w3.org/csswg/css-images-3/#radial-gradients)
In a radial gradient, rather than colors smoothly fading from one side of the gradient box to the other as with linear gradients, they instead emerge from a single point and smoothly spread outward in a circular or elliptical shape.

A radial gradient is specified by indicating the center of the gradient (where the 0% ellipse will be) and the size and shape of the ending shape (the 100% ellipse). Color stops are given as a list, just as for 'linear-gradient()'. Starting from the center and progressing towards (and potentially beyond) the ending shape uniformly-scaled concentric ellipses are drawn and colored according to the specified color stops.

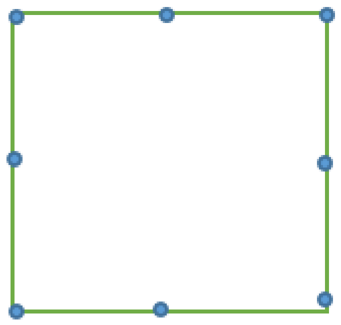
Syntax:
background : radial-gradient(argument1, argument2);

argument1:
In argument1, you provide three values: (1) *ending-shape* (2) *size* and (3) *position*

(1) *ending-shape*: (optional). Can be either 'circle' or 'ellipse'; determines whether the gradient's ending shape is a circle or an ellipse, respectively. If <ending-shape> is omitted, the ending shape defaults to a circle if the <size> is a single <length>, and to an ellipse otherwise.

(2) *size*: (optional). Determines the size of the gradient's ending shape. If omitted it defaults to 'farthest-corner'. It can be given explicitly or by keyword. For the purpose of the keyword definitions, consider the gradient box edges as extending infinitely in both directions, rather than being finite line segments.
If the ending-shape is an ellipse, its axes are aligned with the horizontal and vertical axes.
Both 'circle' and 'ellipse' gradients accept the following keywords as their <size>:
'closest-side'
The ending shape is sized so that that it exactly meets the side of the gradient box closest to the gradient's center. If the shape is an ellipse, it exactly meets the closest side in each dimension.
'farthest-side'
Same as 'closest-side', except the ending shape is sized based on the farthest side(s).
'closest-corner'
The ending shape is sized so that that it passes through the corner of the gradient box closest to the gradient's center. If the shape is an ellipse, the ending shape is given the same aspect-ratio it would have if 'closest-side' were specified.
'farthest-corner'
Same as 'closest-corner', except the ending shape is sized based on the farthest corner. If the shape is an ellipse, the ending shape is given the same aspect ratio it would have if 'farthest-side' were specified.
If <ending-shape> is specified as 'circle' or is omitted, the <size> may be given explicitly as length or percentage.

(3) *position*: Determines the center of the gradient. If this argument is omitted, it defaults to 'center'.
Possible values for *position*: Any of the eight points (they specify the gradient center):

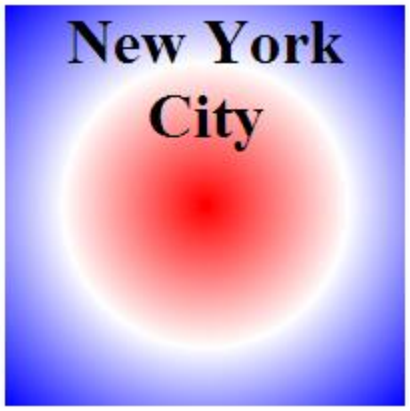


The keywords are:
at top left, **at top right**, **at bottom left, at bottom right, at top center, at right center, at bottom center, at left center**

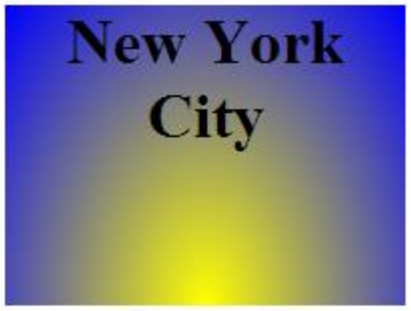
argument2:
This is, as before, color stops.

EXAMPLE:
background: radial-gradient(red, white, blue);

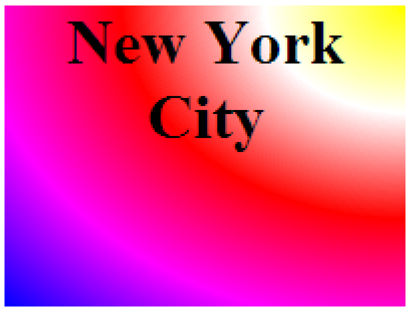
Results in background gradient circular, reaching the farthest-corner with colors red, blue and green (because the element is a rectangle, the gradient is not a perfect circle).



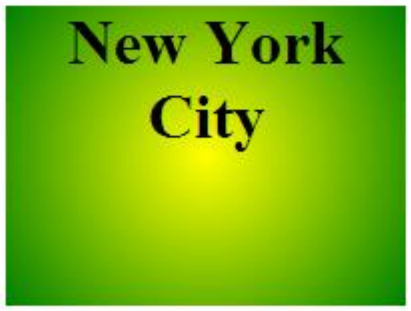
EXAMPLE:
background: radial-gradient(at bottom center, yellow, blue)



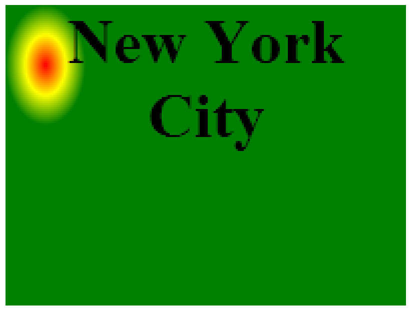
EXAMPLE:
background: radial-gradient(at top right, yellow,white, red, magenta, blue)



EXAMPLE:
background: radial-gradient(circle, yellow, green);



Example:
background: radial-gradient(20px 30px at 20px 30px, red, yellow, green);



---

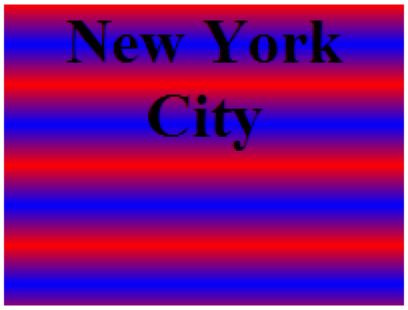
5. Repeating Gradients

In addition to linear-gradient and radial-gradient, we can have repeating gradients – both linear and radial.
The names of the functions are
repeating-linear-gradient()
and
repeating-radial-gradient()

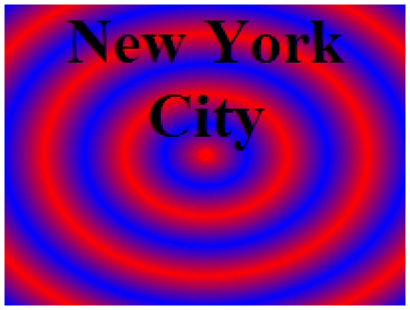
The arguments for these two are similar to the linear-gradient() and radial-gradient() arguments respectively.

In the repeating functions, the color-stops are repeated infinitely in both directions, with their positions shifted by multiples of the difference between the last specified color-stop's position and the first specified color-stop's position.

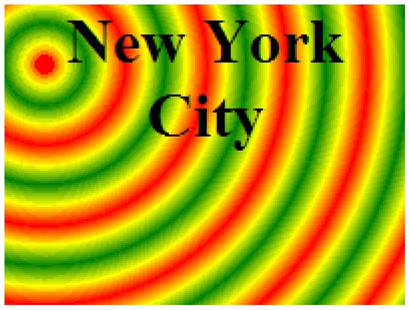
Example:
background: repeating-linear-gradient(red, blue 20px, red 40px);



Example:
background: repeating-radial-gradient(red, blue 20px, red 40px)



Example:
background: repeating-radial-gradient(circle closest-side at 20px 30px, red, yellow, green 100%, yellow 150%, red 200%)



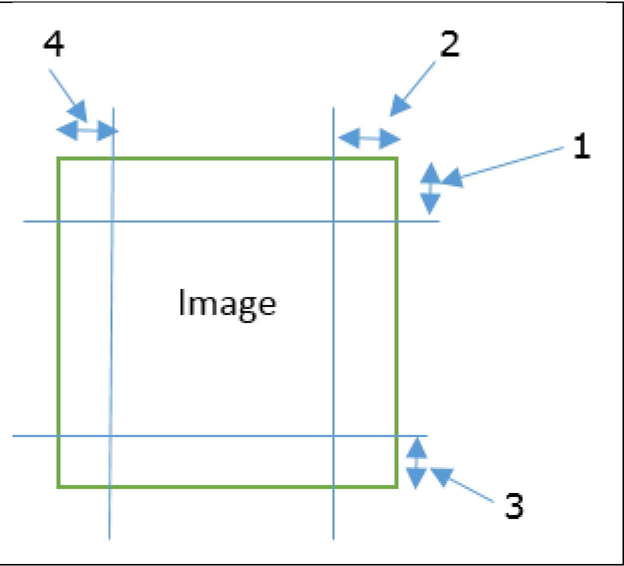
6. The border-image property

The border-image property is used in place of the border styles. In this case, the border's design is taken from the sides and corners of a specified image. The image may be sliced, scaled and stretched in various ways to fit the size of the border image area. The border-image properties do not affect layout: layout of the box, its content, and surrounding content is based on the ‘border-width’ and ‘border-style’ properties only.

Syntax:
We will discuss a syntax with three values:
border-image : *value1 value2 value3;*

value1: border image source
The image for the border, specified in the format url(*imageFilename*).

value2: border image slice
This requires a detailed explanation.
value2 is a numeric value. It represents the thickness of the image from the border.
Let me explain. Imagine the image is divided as follows.

	In the diagram, 1 is the top thickness 2 is the right thickness 3 is the bottom thickness 4 is the left thickness

value2 is represented as a percentage or just a number (it is actually pixels, but px is not used).

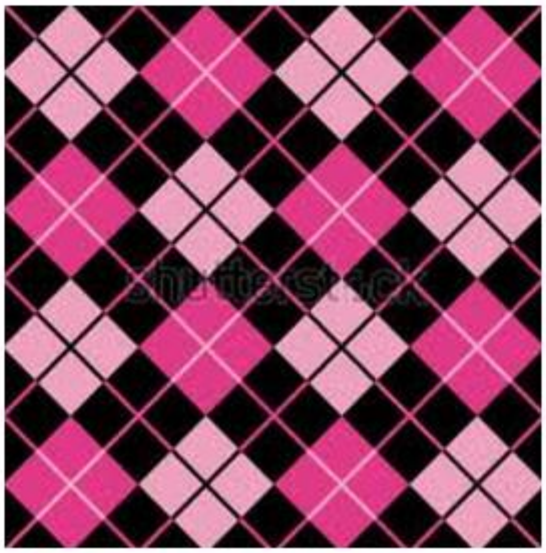
value3 is one these keywords:

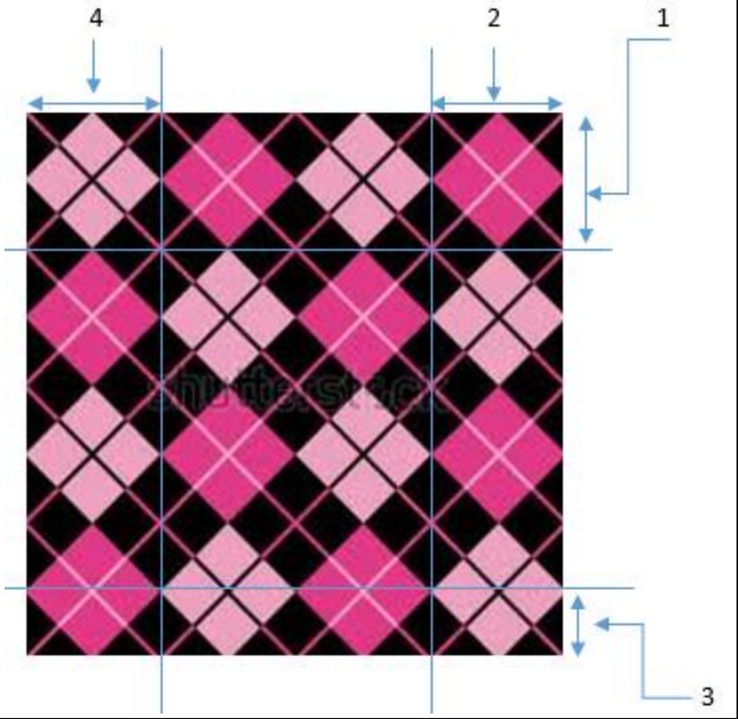
- **repeat**
  The image is tiled (repeated) to fill the area
- **round**
  The image is tiled (repeated) to fill the area. If it does not fill the area with a whole number of tiles, the image is rescaled so that it does.
- **stretch**
  The image is stretched to fill the area.

Now, this is border-image works,
border-image : *value1 value2 value3;*

The portion of the image obtained from *value2* will be placed on the border of the element in the format format specified in *value3*. Corners in the element will be the same as the corners in original image.

EXAMPLE:
Let us take an image and let us draw lines to mark the four thicknesses and also mark percentages of thicknesses:

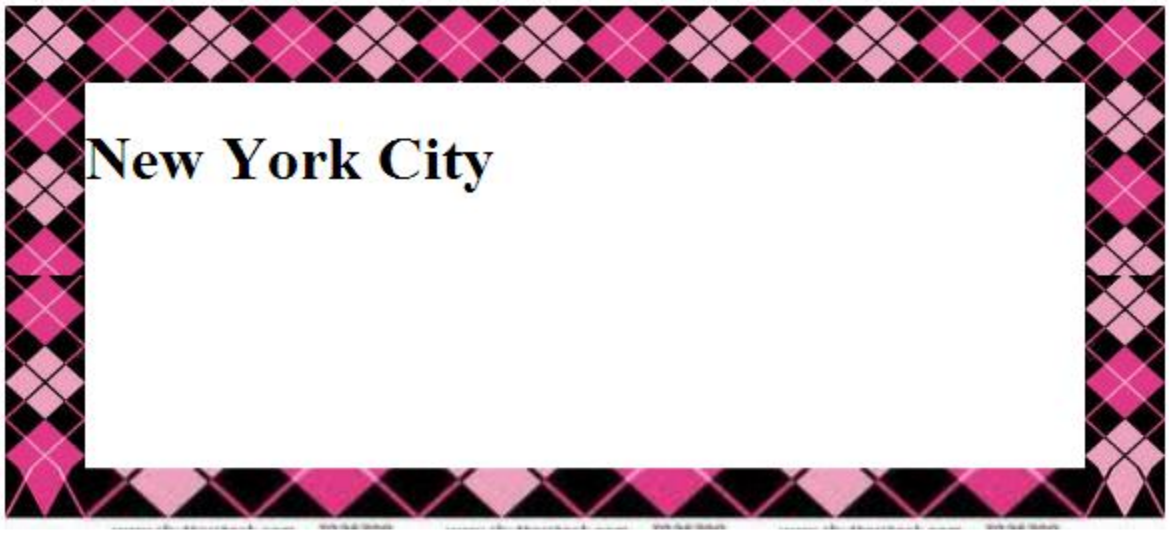




Assume, 1 is 25%, 2 is 25%, 3 is 12.5% and 4 is 25%.

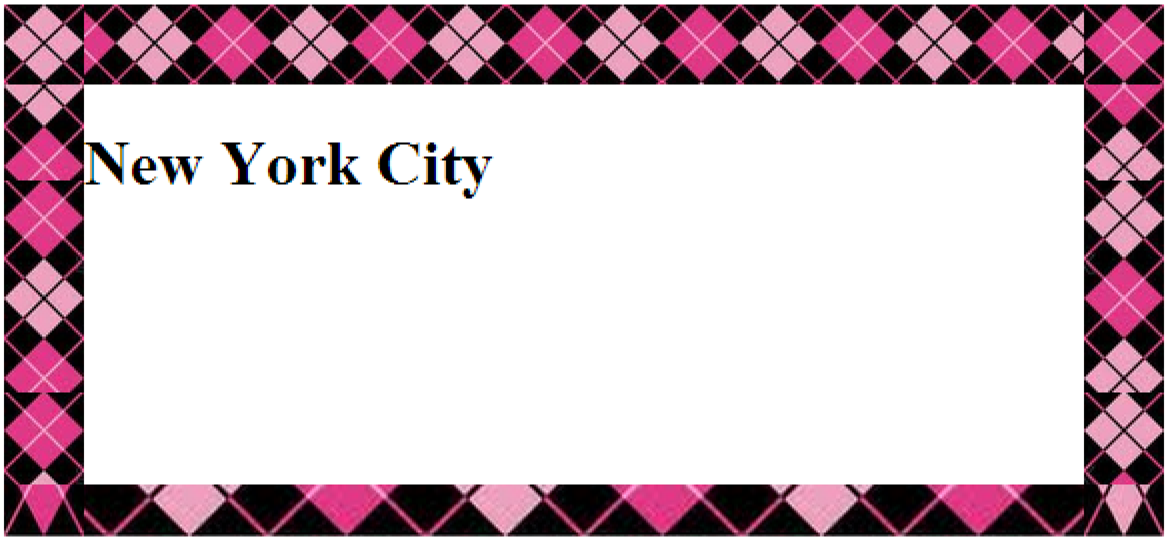
```
<style>
div{
   width:500px;
   height:200px;
   border:40px;
   border-image: url(img2.jpg) 25% 25% 12.5% 25% round  ;
}
</style>
<body>
  <div><h1>New York City</h2> </div>
</body>
```

Result:



Example:

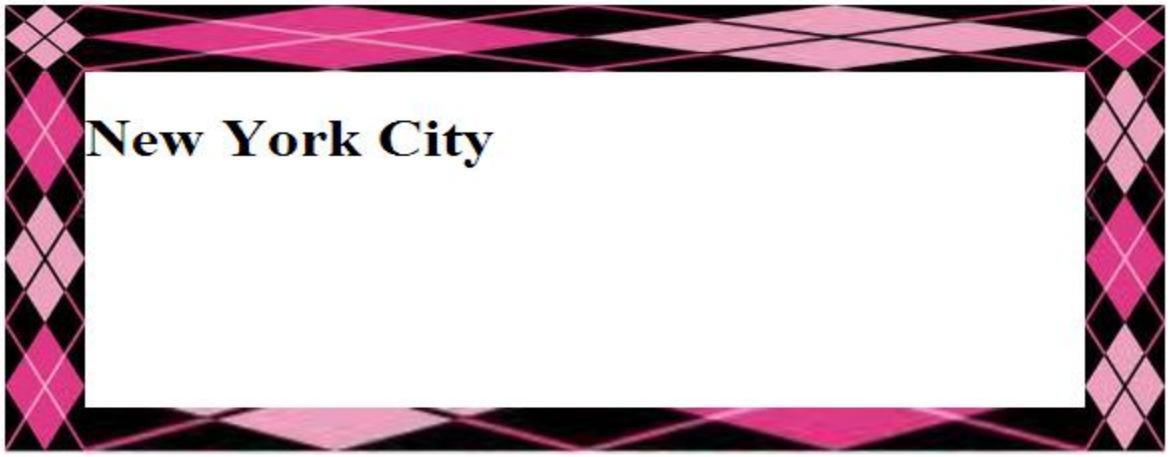
Same example with **border-image: url(img2.jpg) 25% 25% 12.5% 25% repeat;**



Compare these two at the top row, just before the corner diamond.

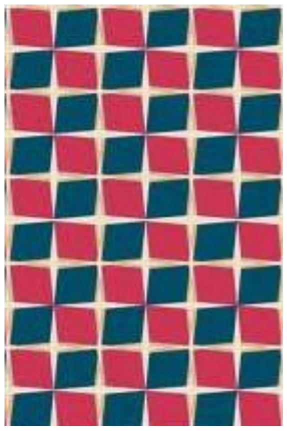
Example:

Same example with **border-image: url(img2.jpg) 25% 25% 12.5% 25% stretch;**

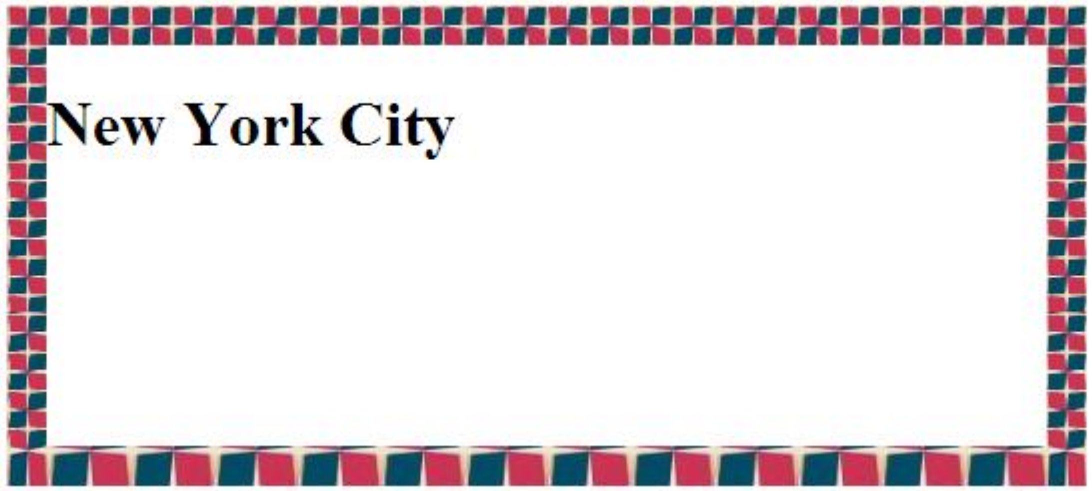


Example.

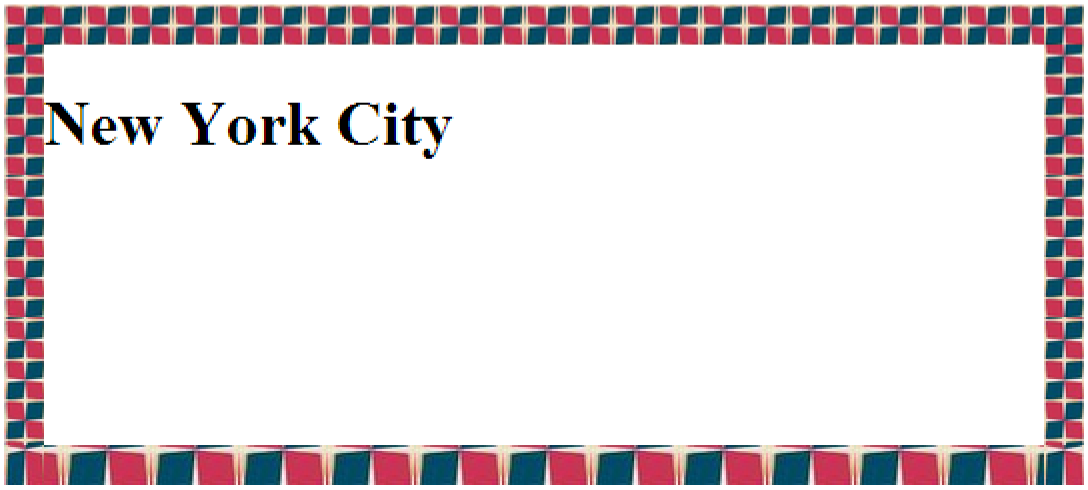
Let us take another example. Original image:



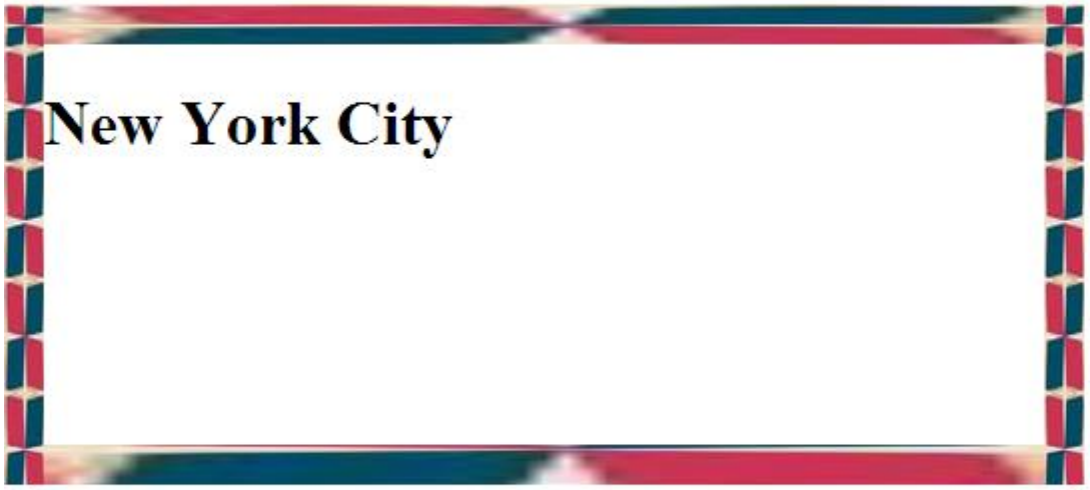
border-image: url(image2.jpg) 20% 33.5% 10% 33.5% round;



border-image: url(image2.jpg) 20% 33.5% 10% 33.5% repeat;



border-image: url(image2.jpg) 20% 33.5% 10% 33.5% stretch;



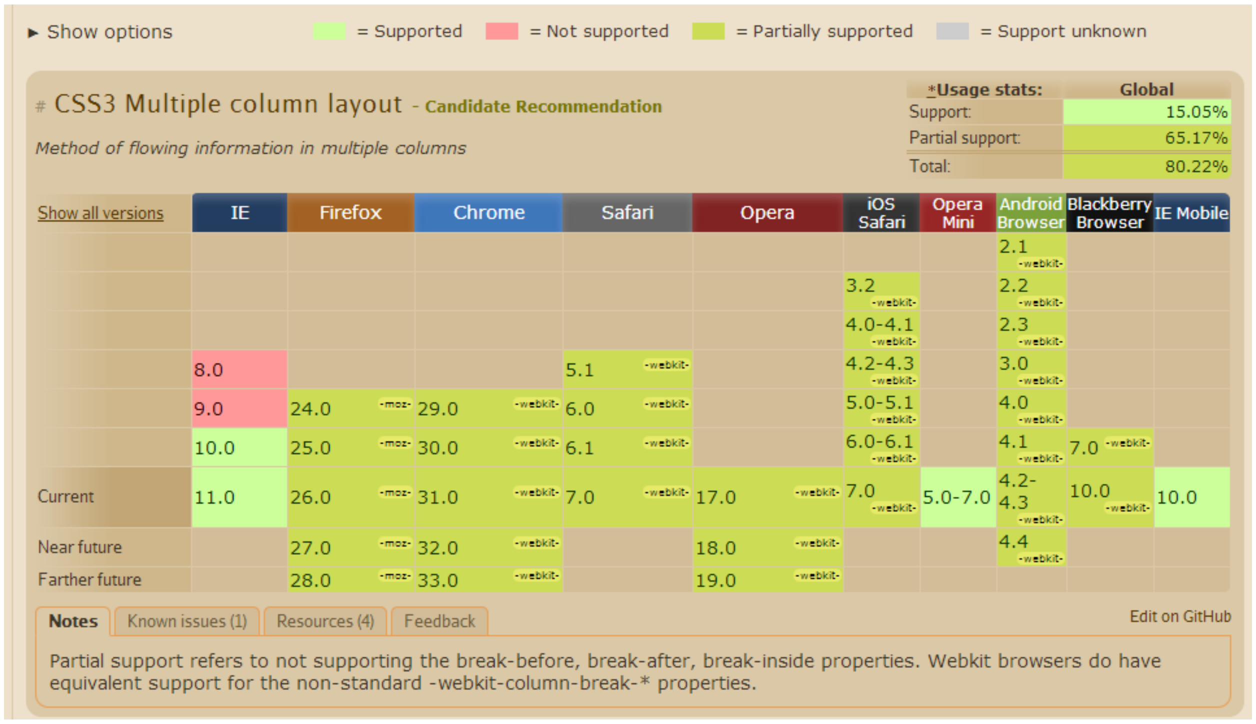
7. Multi-column layout

The CSS column related properties enable the multi-column display to make reading easier than long lines of text, just as newspapers do.

Unfortunately, Chrome and Firefox do not support them directly. IE does. To make columns feature work on Chrome and Firefox, we use their specific prefixes: -webkit- (for Chrome and Safari), and –moz- (for Firefox). What we do in code we include these prefix versions in addition to the standard form.

For the latest stutus of browser support, go to the "caniuse" Web site: http://caniuse.com/#search=column

Here is a display of browser support for columns (as of October 15, 2013).


▸ Show options 🟩 = Supported 🟥 = Not supported 🟨 = Partially supported ⬜ = Support unknown

# CSS3 Multiple column layout - Candidate Recommendation

Method of flowing information in multiple columns

*Usage stats:	Global
Support:	15.05%
Partial support:	65.17%
Total:	80.22%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1 -webkit-		
						3.2 -webkit-		2.2 -webkit-		
						4.0-4.1 -webkit-		2.3 -webkit-		
	8.0			5.1 -webkit-		4.2-4.3 -webkit-		3.0 -webkit-		
	9.0	24.0 -moz-	29.0 -webkit-	6.0 -webkit-		5.0-5.1 -webkit-		4.0 -webkit-		
	10.0	25.0 -moz-	30.0 -webkit-	6.1 -webkit-		6.0-6.1 -webkit-		4.1 -webkit-	7.0 -webkit-	
Current	11.0	26.0 -moz-	31.0 -webkit-	7.0 -webkit-	17.0 -webkit-	7.0 -webkit-	5.0-7.0	4.2-4.3 -webkit-	10.0 -webkit-	10.0
Near future		27.0 -moz-	32.0 -webkit-		18.0 -webkit-			4.4 -webkit-		
Farther future		28.0 -moz-	33.0 -webkit-		19.0 -webkit-					

Notes | Known issues (1) | Resources (4) | Feedback   Edit on GitHub

Partial support refers to not supporting the break-before, break-after, break-inside properties. Webkit browsers do have equivalent support for the non-standard -webkit-column-break-* properties.

CSS provides several properties related to columns display.

➢ **The column-count property**

The column-count property sepecifies the number of columns to display. The column widths will vary depending on the available width.

Syntax:

column-count : *number*;

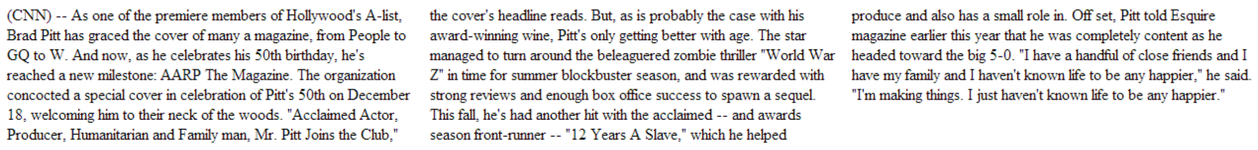
The value *number* is just a number indicating number of columns to display.

EXAMPLE:

```
<style>
#cnn
{
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari and Chrome */
column-count:3;
}
</style>
</head>
<body>
<div id="cnn">
(CNN) -- As one of the premiere members of Hollywood's A-list, Brad Pitt has graced the
cover of many a magazine, from People to GQ to W. And now, as he celebrates his 50th
```

```
birthday, he's reached a new milestone: AARP The Magazine. The organization concocted a
special cover in celebration of Pitt's 50th on December 18, welcoming him to their neck of
the woods." Acclaimed Actor, Producer, Humanitarian and Family man, Mr. Pitt Joins the
Club," the cover's headline reads. But, as is probably the case with his award-winning wine,
Pitt's only getting better with age. The star managed to turn around the beleaguered
zombie thriller "World War Z" in time for summer blockbuster season, and was rewarded
with strong reviews and enough box office success to spawn a sequel. This fall, he's had
another hit with the acclaimed -- and awards season front-runner -- "12 Years
A Slave," which he helped produce and also has a small role in. Off set, Pitt told Esquire
magazine earlier this year that he was completely content as he headed toward the big 5-0.
"I have a handful of close friends and I have my family and I haven't known life to be any
happier," he said. "I'm making things. I just haven't known life to be any happier."
</div>
</body>
```

Result:



(CNN) -- As one of the premiere members of Hollywood's A-list, Brad Pitt has graced the cover of many a magazine, from People to GQ to W. And now, as he celebrates his 50th birthday, he's reached a new milestone: AARP The Magazine. The organization concocted a special cover in celebration of Pitt's 50th on December 18, welcoming him to their neck of the woods. "Acclaimed Actor, Producer, Humanitarian and Family man, Mr. Pitt Joins the Club," the cover's headline reads. But, as is probably the case with his award-winning wine, Pitt's only getting better with age. The star managed to turn around the beleaguered zombie thriller "World War Z" in time for summer blockbuster season, and was rewarded with strong reviews and enough box office success to spawn a sequel. This fall, he's had another hit with the acclaimed -- and awards season front-runner -- "12 Years A Slave," which he helped produce and also has a small role in. Off set, Pitt told Esquire magazine earlier this year that he was completely content as he headed toward the big 5-0. "I have a handful of close friends and I have my family and I haven't known life to be any happier," he said. "I'm making things. I just haven't known life to be any happier."

- **The column-gap property**

The column-gap property controls the gap between columns. The recommended default is 1em. You can change it.

Syntax:

column-gap: *length*;

The *value* is the new gap in the usual length units.

- **The column-rule-width property**

The column-rule-width property specifies the width of the rule between columns.

Syntax:
column-rule-width : *value*;

Value can be one of the keywords, thin, medium (default), or thick. You can also use length in the usual units.

- **The column-rule-style property**

The column-rule-style property specifies the style of the rule between columns.
Syntax:
column-rule-style : *value*;

Value is one of the keywords we have seen before (none, dotted, solid, double, etc.).

- **The column-rule-color property**

The **column-rule-color** property specifies the color of the rule between columns.
Syntax:
column-rule-color : *value*;

Value is one of the color values.

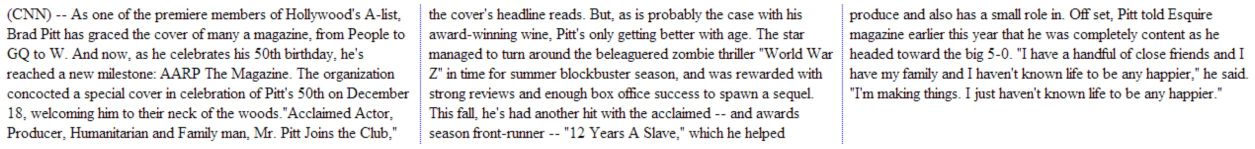
NOTE:
column-rule-style must be set for **column-rule-color** to work.

- **The column-rule property**

The **column-rule** property is the shorthand to declare these three in one line: **column-rule-width, column-rule-style,** and **column-rule-color.**

Syntax:
column-rule : *value1 value2 value3*

value1 is **column-rule-width** value, *value2* is **column-rule-style** value, and *value3* is **column-rule-style**.

EXAMPLE:
```
#cnn
{
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari and Chrome */
column-count:3;
-moz-column-rule: thin dotted blue;
-webkit-column-rule: thin dotted blue;
 column-rule: thin dotted blue;
}
```

(CNN) -- As one of the premiere members of Hollywood's A-list, Brad Pitt has graced the cover of many a magazine, from People to GQ to W. And now, as he celebrates his 50th birthday, he's reached a new milestone: AARP The Magazine. The organization concocted a special cover in celebration of Pitt's 50th on December 18, welcoming him to their neck of the woods."Acclaimed Actor, Producer, Humanitarian and Family man, Mr. Pitt Joins the Club," the cover's headline reads. But, as is probably the case with his award-winning wine, Pitt's only getting better with age. The star managed to turn around the beleaguered zombie thriller "World War Z" in time for summer blockbuster season, and was rewarded with strong reviews and enough box office success to spawn a sequel. This fall, he's had another hit with the acclaimed -- and awards season front-runner -- "12 Years A Slave," which he helped produce and also has a small role in. Off set, Pitt told Esquire magazine earlier this year that he was completely content as he headed toward the big 5-0. "I have a handful of close friends and I have my family and I haven't known life to be any happier," he said. "I'm making things. I just haven't known life to be any happier."

- **The column-span property**

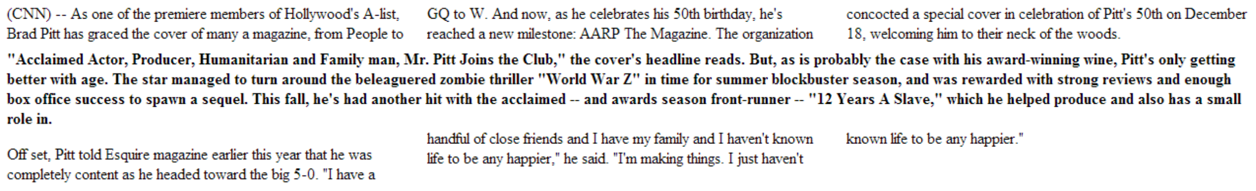
The column-span spans the contents across several columns. A common used value is **all**.

EXAMPLE:
#cnn

```
{
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari and Chrome */
column-count:3;
-moz-column-fill: balance;
-webkit-column-fill: balance;
column-fill: balance;
}
#para
{
-moz-column-span: all;
-webkit-column-span: all;
column-span: all;
}
</style>
</head>
<body>
<div id="cnn">
(CNN) -- As one of the premiere members of Hollywood's A-list, Brad Pitt has graced the
cover of many a magazine, from People to GQ to W. And now, as he celebrates his 50th
birthday, he's reached a new milestone: AARP The Magazine. The organization concocted a
special cover in celebration of Pitt's 50th on December 18, welcoming him to their neck of
the woods.<p> <div id="para"><b>"Acclaimed Actor, Producer, Humanitarian and Family
man, Mr. Pitt Joins the Club," the cover's headline reads. But, as is probably the case with
his award-winning wine, Pitt's only getting better with age. The star managed to turn
around the beleaguered zombie thriller "World War Z" in time for summer blockbuster
season, and was rewarded with strong reviews and enough box office success to spawn a
sequel. This fall, he's had another hit with the acclaimed -- and awards season front-runner
-- "12 Years  A Slave," which he helped produce and also has a small role  n.</b> </div>
<p> Off set, Pitt told Esquire magazine earlier this year that he was completely content as
he headed toward the big 5-0. "I have a handful of close friends and I have my family and I
haven't known life to be any happier,"  he said. "I'm making things. I just haven't known life
to be any happier."
</div>
</body>
```

Notice <div id="para"> within the big paragraph. Also notice CSS style for #para.

Result:



(CNN) -- As one of the premiere members of Hollywood's A-list, Brad Pitt has graced the cover of many a magazine, from People to GQ to W. And now, as he celebrates his 50th birthday, he's reached a new milestone: AARP The Magazine. The organization concocted a special cover in celebration of Pitt's 50th on December 18, welcoming him to their neck of the woods.

"Acclaimed Actor, Producer, Humanitarian and Family man, Mr. Pitt Joins the Club," the cover's headline reads. But, as is probably the case with his award-winning wine, Pitt's only getting better with age. The star managed to turn around the beleaguered zombie thriller "World War Z" in time for summer blockbuster season, and was rewarded with strong reviews and enough box office success to spawn a sequel. This fall, he's had another hit with the acclaimed -- and awards season front-runner -- "12 Years A Slave," which he helped produce and also has a small role in.

Off set, Pitt told Esquire magazine earlier this year that he was completely content as he headed toward the big 5-0. "I have a handful of close friends and I have my family and I haven't known life to be any happier," he said. "I'm making things. I just haven't known life to be any happier."

8. The filter property

The CSS **filter** property provides for interesting visual effects like blurring or color shifting on an element's rendering before the element is displayed. Filters are commonly used to adjust the rendering of an image, a background, or a border.

Browsers do not support filter property yet. You can, however use filter with –webkit- to work in Chrome (and Safari).

There are several values you can use with filter property. To understand everything you should be a professional photographer (I don't understand them. I just see the results. My entire experience is with Picasa – that is it!). Descriptions are from http://www.w3.org/TR/filter-effects/#FilterProperty. My examples are from various places on the Web (http://www.html5rocks.com/en/tutorials/filters/understanding-css/).

We will use the following code in our examples:

```
<style>
img
{
  -webkit-filter: ------;
}
</style>
<body>
  <img src="monaLisa.jpg" width=200 height=250>
  <img src="sunset.jpg" width=200 height=250>
</body>
```

The original two pictures are here:

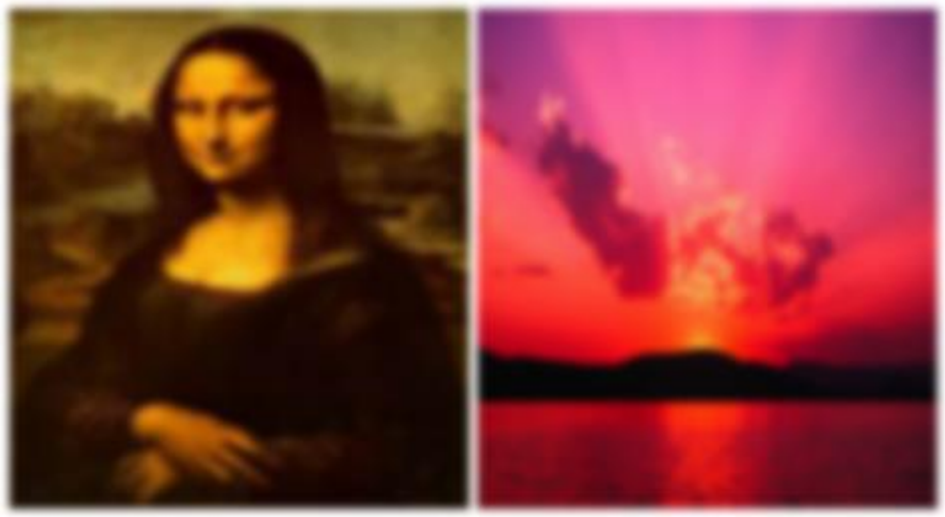
The filter property values take the form of a function. Let us see some of the simple values.

- **blur(*radius*)**

If you want a soft edge for your content, you can add a blur. This one looks like the classic Vaseline on a sheet of glass look that used to be a popular movie making technique. It smudges all the colors together and spreads their effect out - kind of like when your eyes are out of focus. The 'radius' parameter affects how many pixels on the screen blend into each other, so a larger value will create more blur. Zero of course leaves the image unchanged.

Example:
-webkit-filter: blur(2px);

Result:



---

brightness(*amount*)

Applies a linear multiplier to input image, making it appear more or less bright. A value of 0% will create an image that is completely black. A value of 100% leaves the input unchanged. Other values are linear multipliers on the effect. Values of amount over 100% are allowed, providing brighter results.

Example:
-webkit-filter: brightness(150%);

- **contrast()**

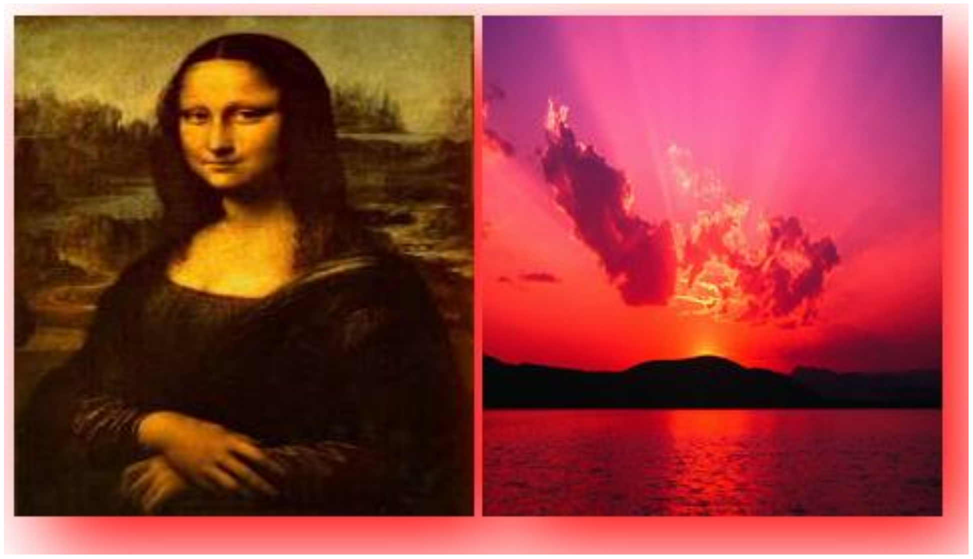
Adjusts the contrast of the input. A value of 0% will create an image that is completely gray. A value of 100% leaves the input unchanged. Values of amount over 100% are allowed, providing results with less contrast.

EXAMPLE:
-webkit-filter: contrast(150%);

- **drop-shadow()**

Applies a drop shadow effect to the input image. A drop shadow is effectively a blurred, offset version of the input image's alpha mask drawn in a particular color, composited below the image. The argument values are similar to box-shadow values.

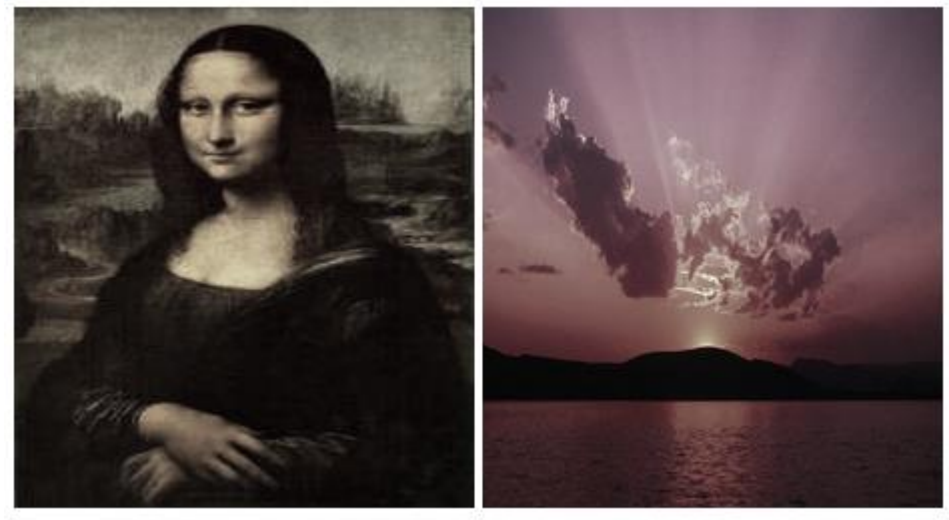
Example:
-webkit-filter: drop-shadow(15px 15px 20px red);



- **grayscale()**

Converts the input image to grayscale. The passed parameter defines the proportion of the conversion. A value of 100% is completely grayscale. A value of 0%leaves the input unchanged. Values between 0% and 100% are linear multipliers on the effect.

EXAMPLE:
-webkit-filter:grayscale(80%);

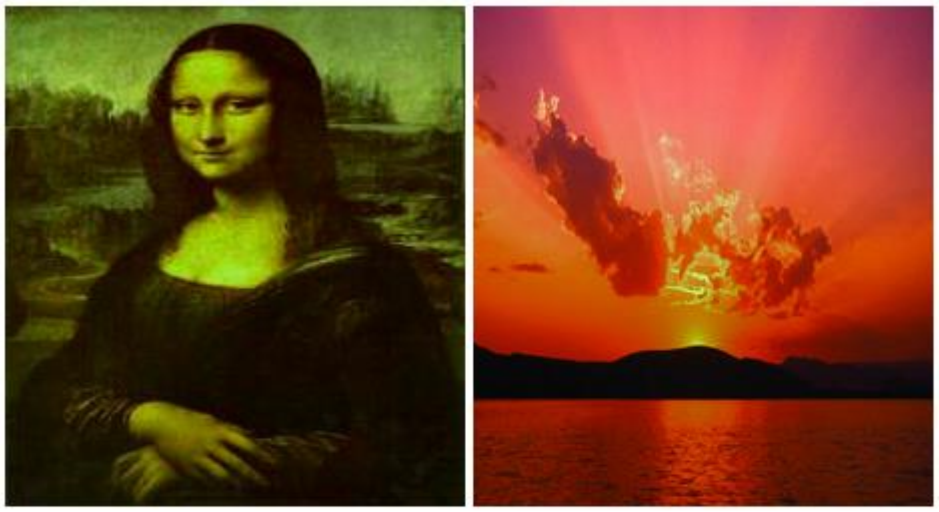


- **hue-rotate()**

Applies a hue rotation on the input image. The passed parameter defines the number of degrees around the color circle the input samples will be adjusted. A value of 0deg leaves the input unchanged.

EXAMPLE:

-webkit-filter:hue-rotate(30deg);

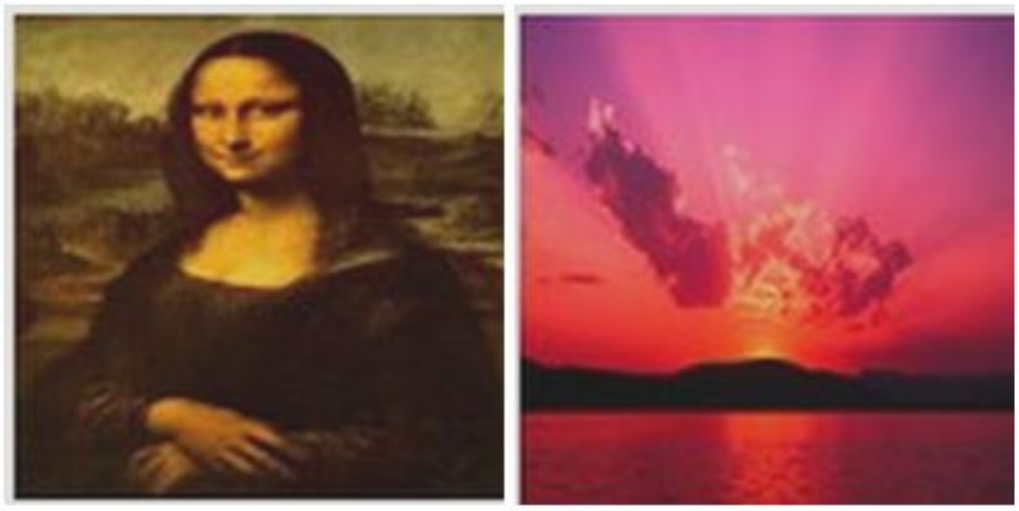


- **invert()**

Inverts flips the colors in input image. The passed parameter defines the proportion of the conversion. A value of 100% is completely inverted. A value of 0% leaves the input unchanged. Values between 0% and 100% are linear multipliers on the effect.

EXAMPLE:

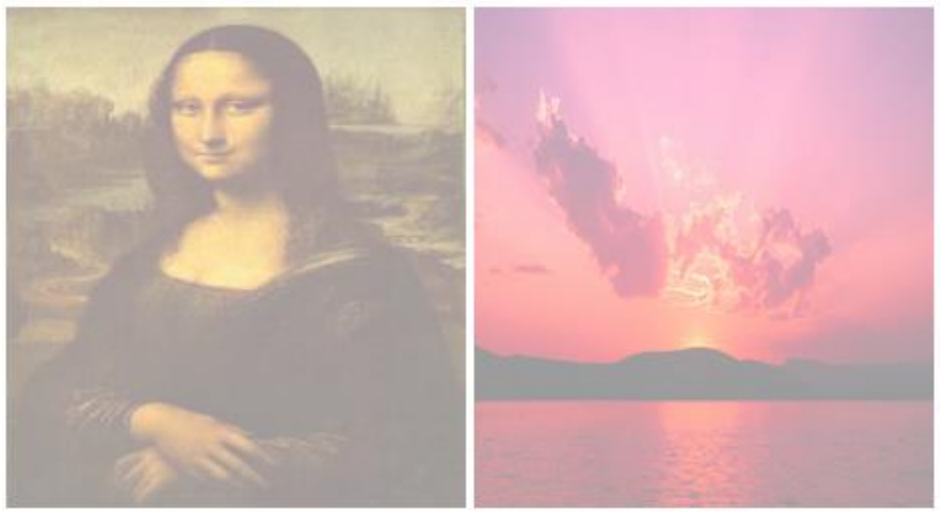
-webkit-filter: invert(10%);



- **opacity()**

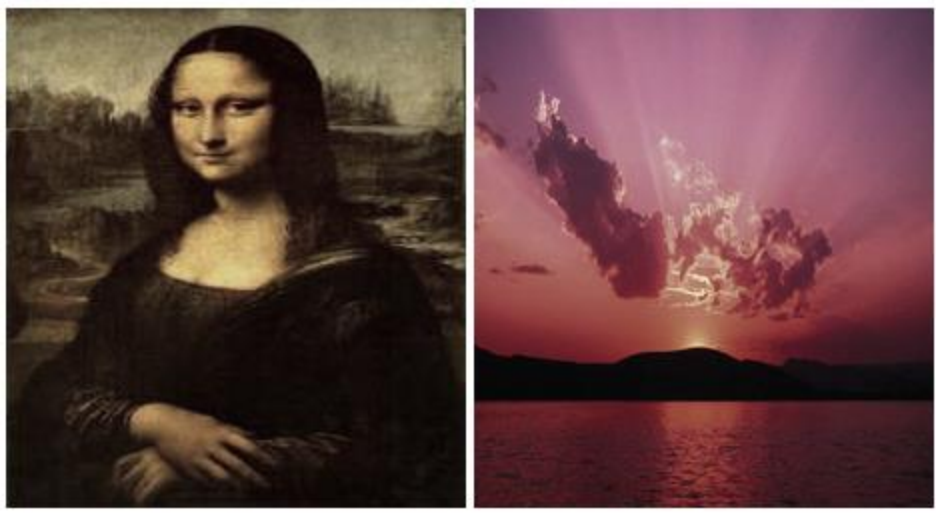
Applies transparency to the samples in the input image. The passed parameter defines the proportion of the conversion. A value of 0% is completely transparent. A value of 100% leaves the input unchanged. Values between 0% and 100% are linear multipliers on the effect.

EXAMPLE:
-webkit-filter:opacity(40%);



➢ **saturate()**
Saturates the input image. The passed parameter defines the proportion of the conversion. A value of 0% is completely un-saturated. A value of 100% leaves the input unchanged. Other values are linear multipliers on the effect. Values of amount over 100% are allowed, providing super-saturated results.

EXAMPLE:
-webkit-filter:saturate(40%);

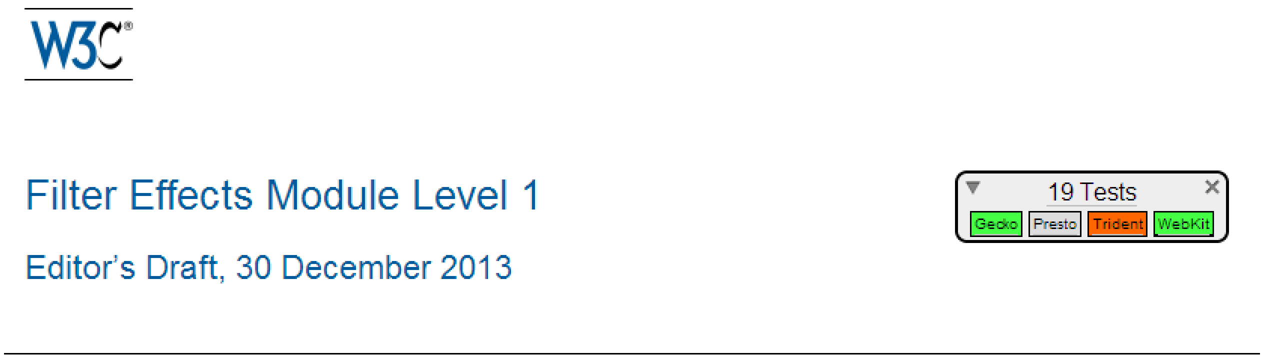


➢ **sepia()**
Converts the input image to sepia (produces a tinge like in old photographs). The passed parameter defines the proportion of the conversion. A value of 100% is completely sepia. A value of 0% leaves the input unchanged. Values between 0% and 100% are linear multipliers on the effect.

EXAMPLE:
-webkit-filter:sepia(80%);

NOTE:
You can include more than one value in a single filter property.
For example, -webkit-filter:sepia(80%) blur(2px);
Try this site for hands-on experiments: http://css3.bradshawenterprises.com/filters/

NOTE:
Some of the filter features work in other browsers. See the latest status on this page:
https://dvcs.w3.org/hg/FXTF/raw-file/tip/filters/index.html



Filter Effects Module Level 1

Editor's Draft, 30 December 2013

9. The transform property

Using the CSS **transform** property elements can be translated, rotated, scaled, and skewed according to the values set.

Browsers do not support filter property yet. You can, however use filter with –webkit- to work in Chrome (and Safari). Use –moz- to work with Firefox. Someday soon you will not need these prefixes.

➢ **translateX(*xdistance*)**

Translates the element by the given amount along the X axis. The argument *xdistance* can be length or percentage.

EXAMPLE:
-webkit-transform:translateX(20px);

➢ **translateY(*ydistance*)**

Translates the element by the given amount along the Y axis. The argument *ydistance* can be length or percentage.

EXAMPLE:
-webkit-transform:translateY(20px);

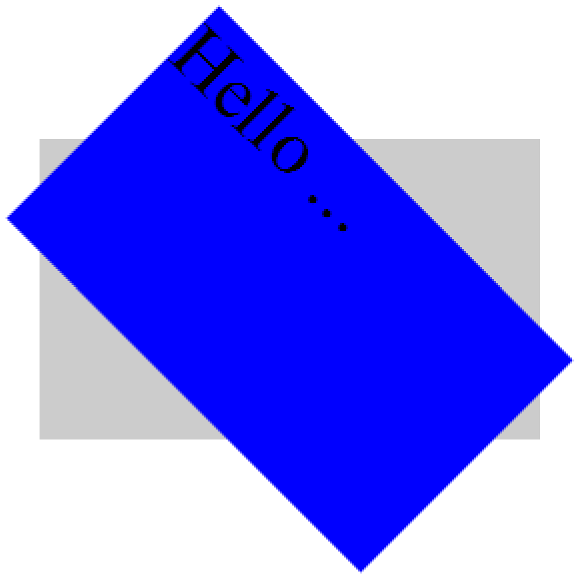
➢ **translate(*xdistance,ydistance*)**
Translates the element by the given amounts along x and y directions . The arguments can be length or percentage.

EXAMPLE:
-webkit-transform:translate(20px, 30%);

- **rotate(*angle*)**

Rotates the element clockwise around its origin. By default the origin is at the center of the element. We can change the origin by using the CSS property transform-origin (see below).

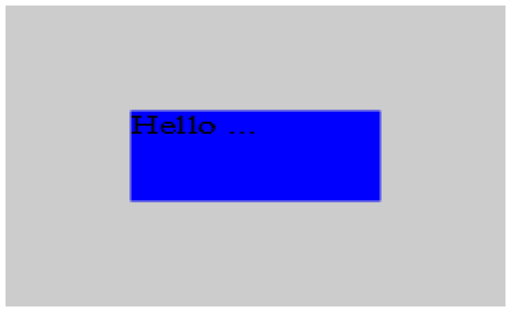
EXAMPLE:
-webkit-transform : rotate(45deg);



- **scale(*value1, value2*)**

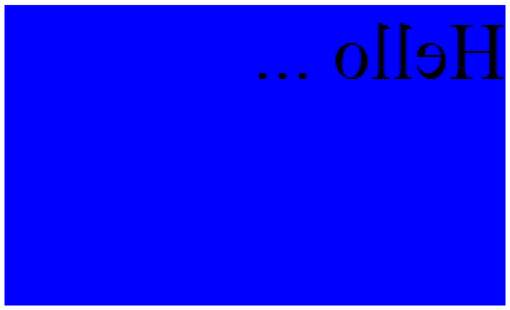
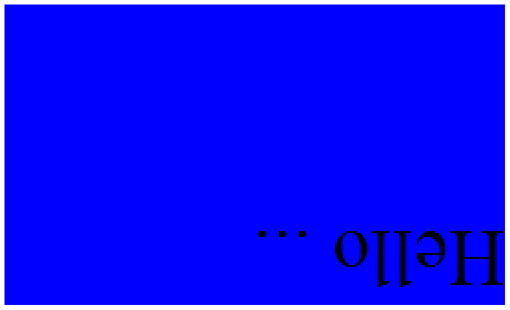
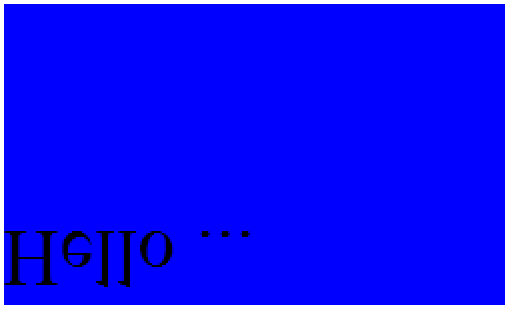
The arguments are two unit less numbers. Specifies a 2D scaling (change in size) operation described by [*value1, value2*]. If *value2* isn't specified, it is assumed to be equal to *value1*.

EXAMPLE:
-webkit-transform : scale(.5,.3);



NOTE:
Other variations of scaling:
scaleY(.3) or scale(1,.3) scale only in Y direction.
scaleX(.25) or scale(.25,1) scale only in X direction.

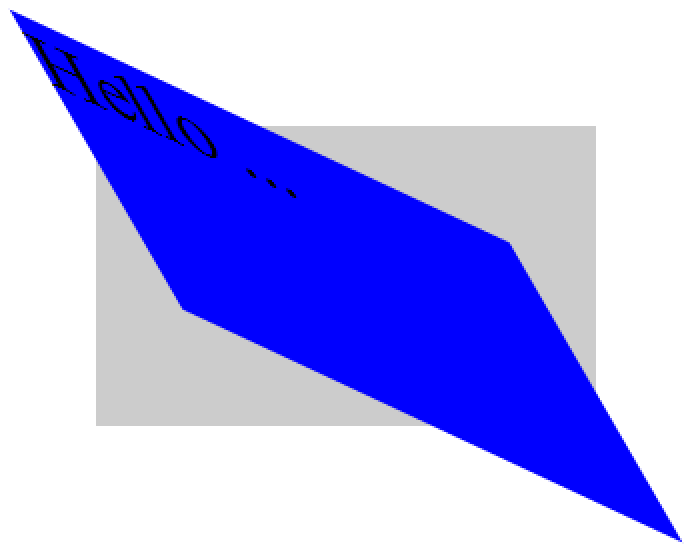
EXAMPLES:
Funny result can be obtained using scale:

-webkit-transform : scale(-1,1);	
-webkit-transform : scale(-1,-1);	
-webkit-transform : scale(1,-1);	

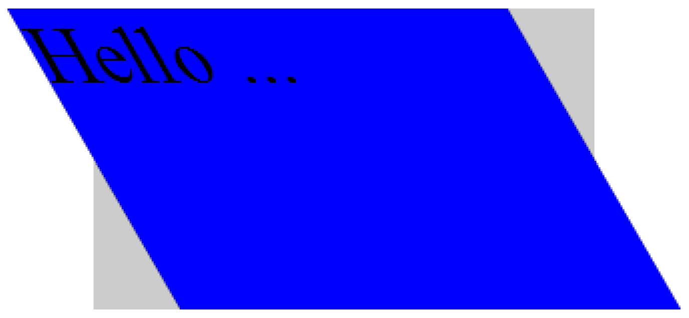
- **skew(value1,value2)**

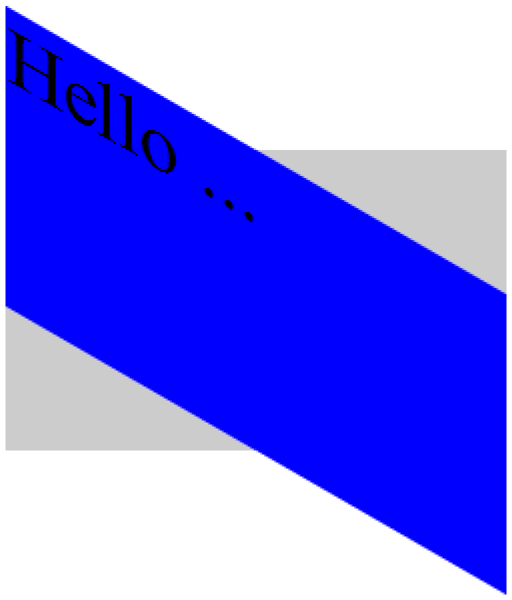
The arguments *value1* and *value2* are angles. Skews the element horizontally by angle given by *value1* angle and vertically by angle *value2*.

EXAMPLE:
-webkit-transform : skew(30deg, 25deg);



Note:
Other variations of skewing:

-webkit-transform : skewX(30deg); or -webkit-transform : skew(30deg,0); Skew only horizontally	

-webkit-transform : skewY(30deg); or -webkit-transform : skew(0,30deg); Skew only vertically	

➢ **The transform-origin property**

The transform-origin CSS property lets you modify the origin for transformations of an element.

By default the origin of an element is the center of the element rectangle (center horizontally and vertically). If you, for example, apply a rotation, the rotation happens around this center. If you like to apply rotation around a different point, you have to first move the center to that point and then apply rotation.

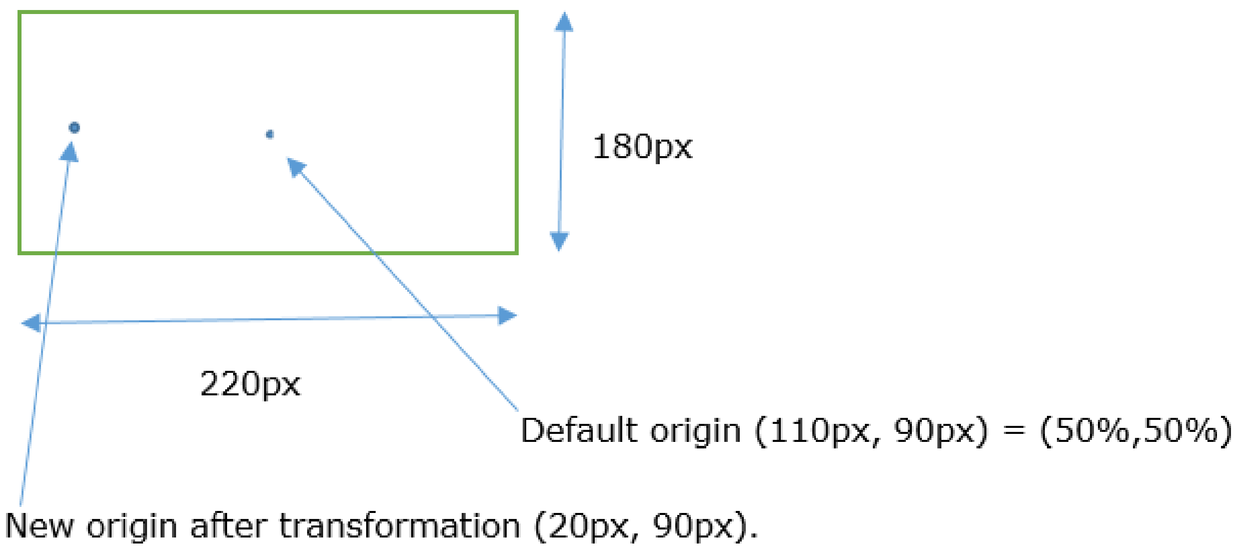
The following are some of the values you can use with the **transform-origin** property: (As before you need to use proper prefixes for this to work)

- **-webkit-transform-origin : *value1*;**

The value1 is length in the usual units or a percentage. This value specifies the new x distance of the center.

EXAMPLE:
Consider an element of size width 220px and height 180px.

-webkit-transform-origin : 20px;

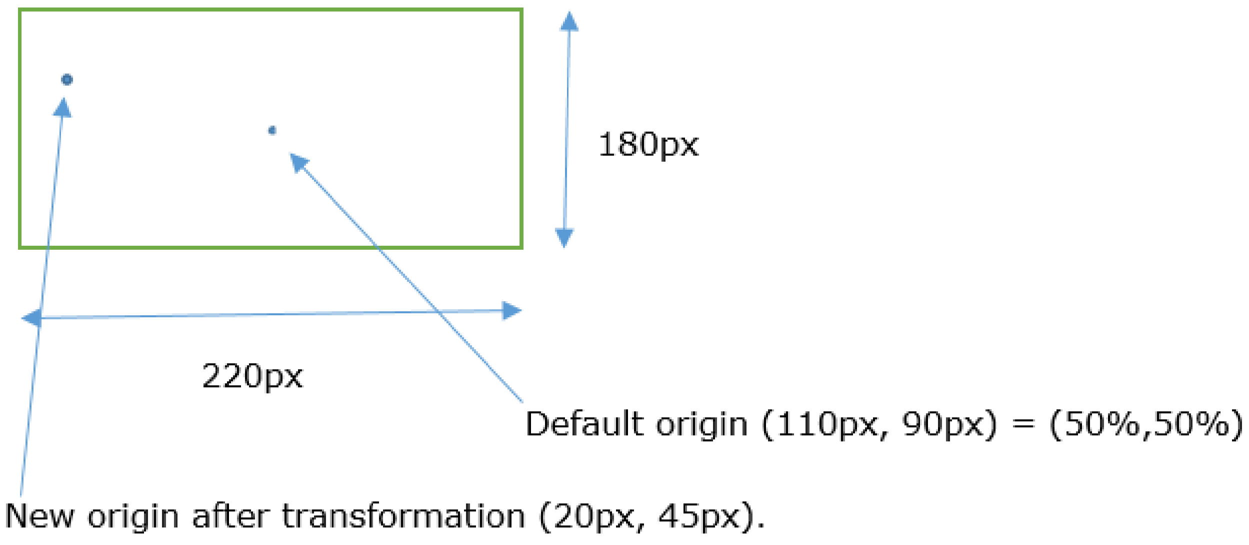

180px
220px
Default origin (110px, 90px) = (50%,50%)
New origin after transformation (20px, 90px).

- **-webkit-transform-origin : *value1 value2*;**

The *value1* and *value2* are lengths in the usual units or a percentages. *Value1* specifies the new x distance and *value2* new y distance.

EXAMPLE:
Consider an element of size width 220px and height 180px.

-webkit-transform-origin : 20px 45px;


180px
220px
Default origin (110px, 90px) = (50%,50%)
New origin after transformation (20px, 45px).

- **Values can be keywords for horizontal offset: left, right or center**
- **Values can be keywords for vertical offset: top, bottom or center**

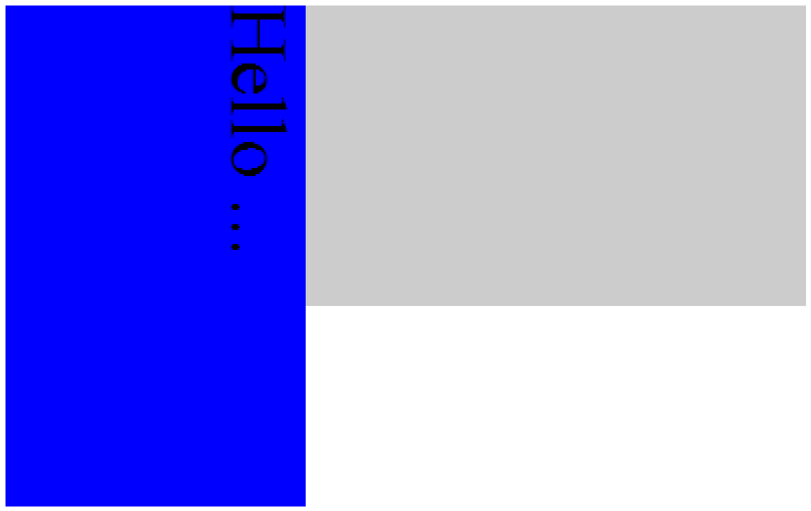
The meanings of these offsets are obvious:

-webkit-transform-origin : left top;
-webkit-transform-origin : right top;
-webkit-transform-origin : center top;
And so on.

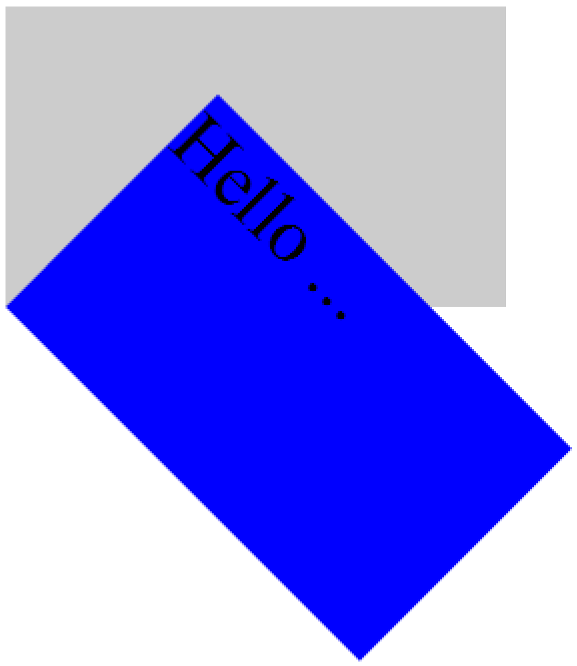
Examples of rotation with transform-origin

EXAMPLE:

-webkit-transform: rotate(90deg);
-webkit-transform-origin : left top;

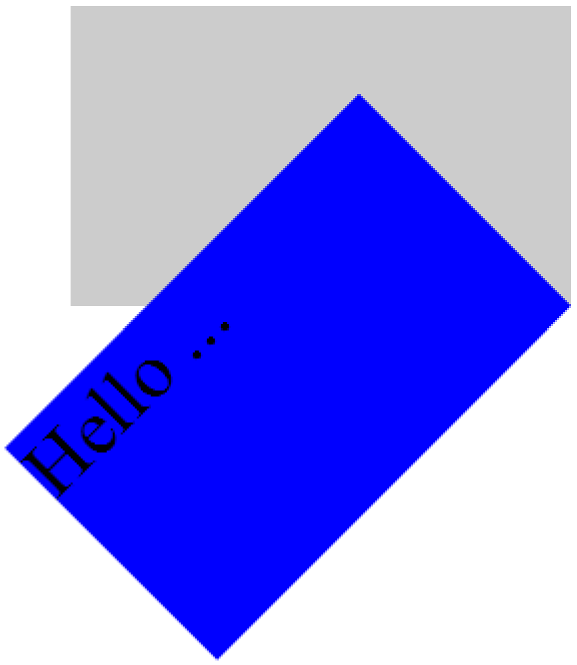


EXAMPLE:
-webkit-transform: rotate(45deg);
-webkit-transform-origin : left bottom;



EXAMPLE:

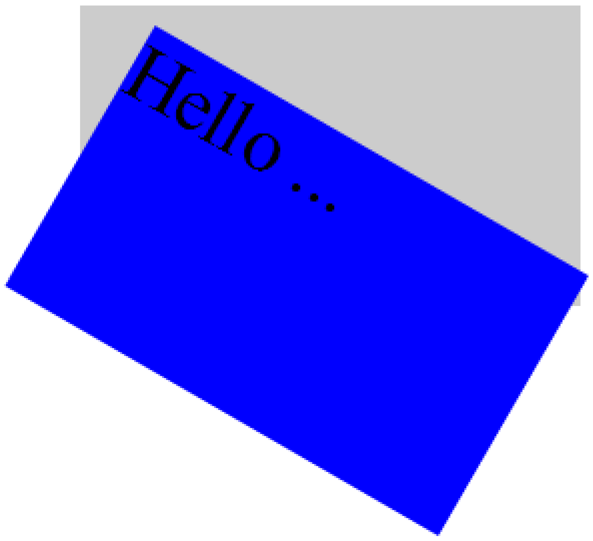
```
-webkit-transform: rotate(-45deg);
-webkit-transform-origin : right bottom;
```



EXAMPLE:

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

-webkit-transform-origin : left center;



NOTE:
You can use transform with several features:
-webkit-transform: translateX(10px) rotate(10deg) translateY(5px)

---

10. Transformations in pseudo classes

A CSS pseudo-class is a keyword added to selectors that specifies a special state of the element to be selected.

Syntax:
selector:pseudo-class{
  CSS properties;
}

CSS3 provides several pseudo-class keywords (see: https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes). We will discuss two: **:active** and **:hover**. They are generally used with <a> tags. We will use it with <div>.

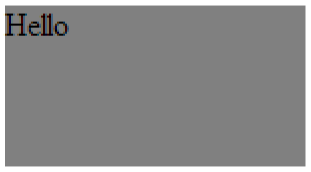
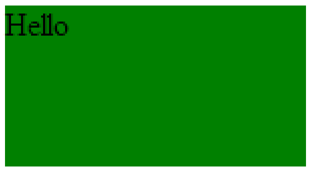
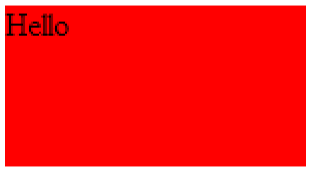
selector:**hover**
This is how it works, when the user hovers over the element specified by the selector, the CSS properties specified in selector**:hover** will apply a style.

selector:**active**
Similarly, when the element is active (that is, user presses mouse button on the element), the CSS properties specified in *selector:***active** will apply a style.

EXAMPLE:

```
<html>
<head>
<style>
div{
  width:150px;
  height:80px;
  background-color:gray;
  }
  div:hover { background-color: green; }
  div:active { background-color: red; }
</style>
</head>
<body>
<div>Hello</div>
</body>
</html>
```

RESULT:

	div:hover	**div:active**
		
	When bring mouse pointer on the element	When press mouse pointer on the element

NOTE:
Pseudo class **active** must come after **hover**.

11. Transitions

Normally when the value of a CSS property changes, the rendered result is instantly updated, with the affected elements immediately changing from the old property value to the new property value. The CSS property **transitions** provide a way to control the speed in which a CSS property change takes place. Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time. For example, if you change the color of an element from white to black, usually the change is instantaneous. With CSS transitions enabled, changes occur at time intervals that follow an acceleration curve, all of which can be customized.
(https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_transitions)
(http://www.w3.org/TR/css3-transitions/)

Not all the properties can be used in creating transitions. See a list here: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties?redirectlocale=en-US&redirectslug=CSS%2FCSS_animated_properties

When you use **transition** property, you need to specify

transition-property
Specifies the name or names of the CSS properties to which transitions should be applied. Only properties listed here are animated during transitions; changes to all other properties occur instantaneously as usual.

The value is a list of properties to be transitioned, or the keyword '**all'** which indicates that all properties are to be transitioned.

transition-duration
Specifies the duration over which transitions should occur. You can specify a single duration that applies to all properties during the transition, or multiple values to allow each property to transition over a different period of time.

The value is a number in seconds.

transition-timing-function
This specifies the rate of change of a parameter over time.
Some of keywords used for value are: **ease, linear, ease-in, ease-out, and ease-in-out.** You have learn how they work by experimenting.

transition-delay
Defines how long to wait between the time a property is changed and the transition actually begins.

The value is a number in seconds.

These concepts will be clear with examples. In our examples, we see both transitions and pseudo classes.

Typical syntax:

transition: *property duration timing-function delay;*

Example

```
div{
  width:200px;
```

```
   height:80px;
   background: blue;
   transition: background-color 2s linear  1s;
}
div:hover {
  background-color: green;
}
```

Originally <div> element is blue. The div{ } CSS rule states that transition is to be applied to background-color property, change takes 2 seconds to complete, transition is linear (smooth), and transition starts after 1 second.

The div:hover{ } states that the color changes to green when mouse hovers on the element.
Compelete code:

```
<html>
<head>
<style>
div{
   width:200px;
   height:80px;
   background: blue;
   transition: background-color 2s linear  1s;
}
div:hover {
  background-color: green;
}
</style>
</head>
<body>
<div>
Hello
</div>
</body>
</html>
```

Example:

Two properties are included in the transition:

```
div{
   width:200px;
   height:80px;
   background: blue;
   transition: font-size background-color 2s linear  1s;
}
div:hover {
  background-color: green;
  font-size:40px;
}
```

Example:

```
div{
   width:100px;
   height:100px;
   background: blue;
   transition: all 2s linear  1s;
}
div:hover {
  background-color: green;
  font-size:40px;
  -webkit-transform:rotate(-360deg);
}
```

12. 3D transformations

(http://www.w3.org/TR/css3-transforms/)
2D transformations happen in a flat plane. Two-dimensional transform functions can alter the appearance of an element, but that element is still rendered into the same plane as its containing block.

Three-dimensional transforms give an illusion of a depth, with the Z axis, which appears to project out of the plane of the screen.

To add a 3-d effect on a transformation, CSS3 provides a property called **perspective**. To understand perspective completely involves matrix algebra. We will not go into the math of it, but just see how to use the property.

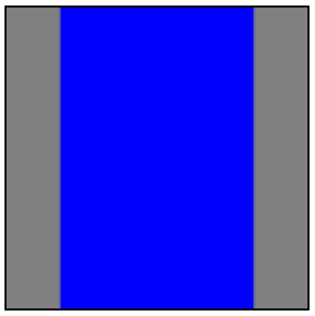
Syntax for using perspective (in Chrome):

-webkit-perspective: *length***;**

Where length is a length in usual units.

EXAMPLE:

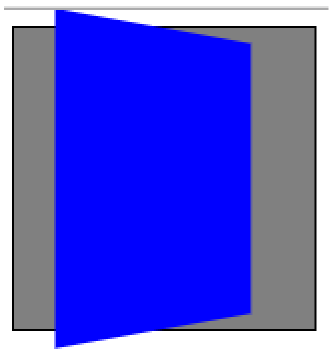
HTML	CSS	Result

`<body>`
`<div class="one">`
`  <div class="two">`
`  </div>`
`</div>`
`</body>`
	`<style>`
	`div {`
	`  height: 150px;`
	`  width: 150px;`
	`}`
	`.one {`
	`  border: 1px solid black;`
	`  background-color:gray;`
	`}`
	`.two {`
	`  -webkit-transform: rotateY(50deg);`
	`   background-color:blue;`
	`}`
	`</style>`
		

This is just a two dimensional rotation on Y axis. The result is the blue box appear narrower, but not there is no three-dimensional effect. Let us now add **perspective** property.

The same code with one additional line is .0ne{ }:
```
.one {
  border: 1px solid black;
  background-color:gray;
  -webkit-perspective: 500px;
}
```

RESULT:



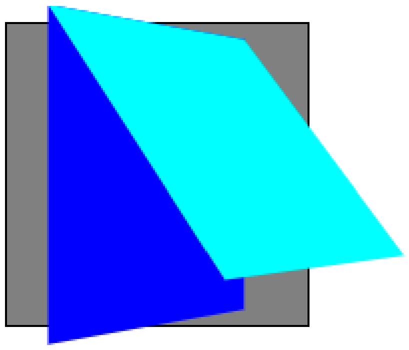
The **perspective** properties can be used to add a feeling of depth to a scene by making elements higher on the Z axis (closer to the viewer) appear larger, and those further away to appear smaller. The **perspective** property is applied on the parent element.

- **The transform-style property**

We use this property with value preserve-3d, which will preserve the 3d effect. The following example shows how to use it.

```
<html><head>
<style>
div {
  height: 150px;
  width: 150px;
}
.one {
  border: 1px solid black;
  background-color:gray;
  -webkit-perspective: 500px;
}
.two {
  -webkit-transform-style: preserve-3d;
  -webkit-transform: rotateY(50deg);
   background-color:blue;
}
.three{
    -webkit-transform-origin: top left;
    -webkit-transform : rotateX(40deg);
    background-color : cyan;
}
</style>
</head>
<body>
<div class="one">
  <div class="two">
    <div class="three">
    </div>
  </div>
</div>
</body>
</html>
```

Result:



Notice the property:

```
  -webkit-transform-style: preserve-3d;
```
Without this, the result would be flat.

Note:

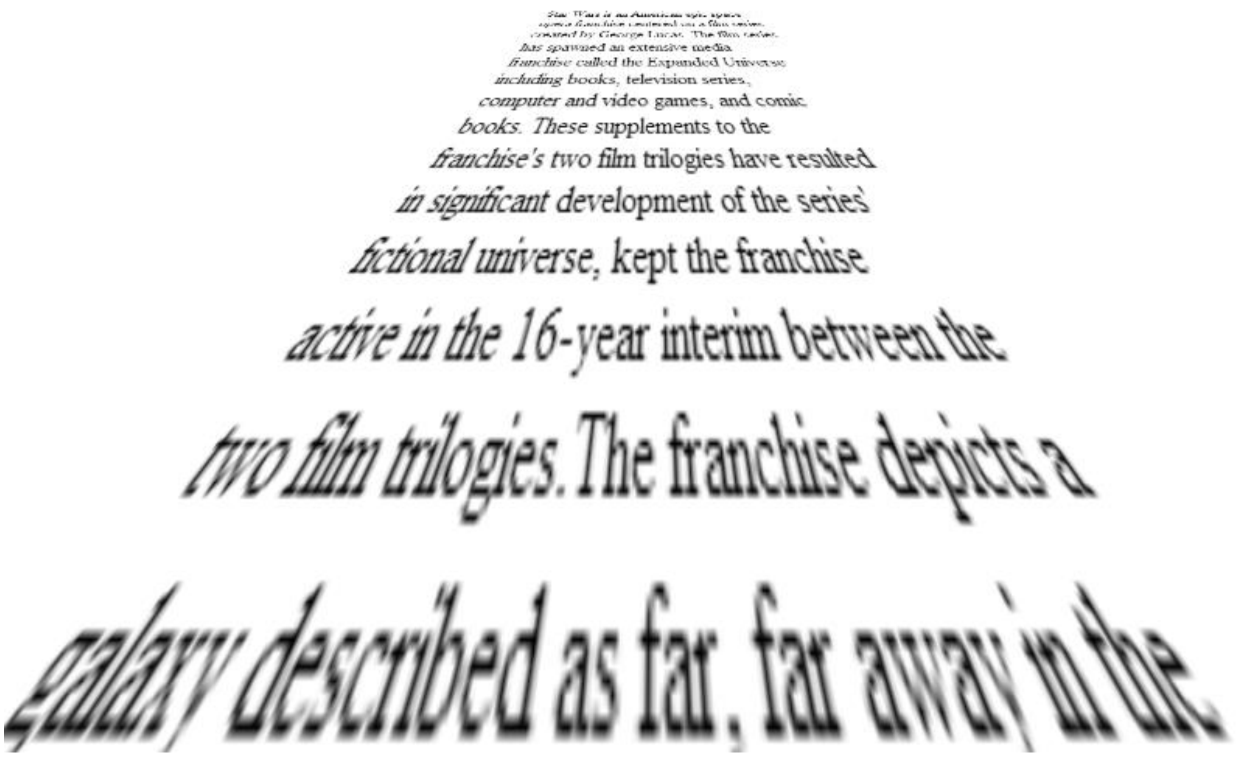
The transform property can be used as follows:
`-webkit-transform: perspective(500px)  scaleZ(2)  rotateX(45deg);`

The property backface-visibility: hidden.
is used to hide the reverse sides of the faces when they are faced away from the viewer.
Does not work well.

Example:
```
<html>
<head>
<style>
.one {
 height: 150px;
  width: 250px;
  margin-left:500px;
  -webkit-perspective: 500px;
}
.two {
  background-color: white;
  -webkit-transform: perspective(500px) scaleZ(2) rotateX(45deg);
}
</style>
</head>
<body>
<div class="one">
```

```
  <div class="two">
      Star Wars is an American epic space opera franchise centered on a
      film series created by George Lucas. The film series has spawned an
      extensive media franchise called the Expanded Universe including books,
      television series, computer and video games, and comic books. These supplements
      to the franchise's two film trilogies have resulted in significant development of the
      series' fictional universe, kept the franchise active in the 16-year interim between the
      two film trilogies. The franchise depicts a galaxy described as far, far away in the
      distant past,
  </div>
</div>
</body>
</html>
```

Result:



Star Wars is an American epic space
opera franchise centered on a film series
created by George Lucas. The film series
has spawned an extensive media
franchise called the Expanded Universe
including books, television series,
computer and video games, and comic
books. These supplements to the
franchise's two film trilogies have resulted
in significant development of the series'
fictional universe, kept the franchise
active in the 16-year interim between the
two film trilogies. The franchise depicts a
galaxy described as far, far away in the

Example:
When you hover the mouse pointer, the picture changes.

```
<!doctype html>
<html>
<head>
   <title>Image Flip Effect with CSS3 3D Transform</title>
<style>
.main
```

```
{
   width: 500px;
   margin: 50px auto;
}
h1
{
   text-align: center;
}

#twoPics
{
  margin-top: 50px;
 -webkit-transform-style: preserve-3d;
}

.picture1, .picture2
{
   padding: 10px;
   width: 500px;
   height: 250px;
   position: absolute;
}

#twoPics:hover .picture1
{
   -webkit-transform: rotateY(180deg);
}
</style>
</head>
<body>
   <div class="main">
      <h1> Flip Pictures with CSS3</h1>
      <div id="twoPics">
         <div class="picture1">
            <img src="pyramids.jpg" alt="pyramids" width=500 height=250/>
         </div>
         <div class="picture2">
            <img src="Sunset.jpg" alt="Sunset" width=500 height=250 />
         </div>
      </div>
   </div>
</body>
</html>
```

Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter3/switchImages.html

EXAMPLE:

This example switches images by smooth rotation.
Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter3/rotateImages/

Here is the HTML file:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter3/rotateImages/rotateImageshtml.pdf

CSS file:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter3/rotateImages/rotateImagesCSS.pdf

---

13. CSS3 Animation

We have already seen how to accomplish some animation using transitions, which smoothly change CSS properties over time. But we can do much more using a CSS feature called keyframes. We can do animation using flash or JavaScript. But for simple animations using CSS is much better than other methods.

CSS Animations are defined by the World Wide Web Consortium (W3C) – the first link above.

---

14. Keyframes

Keyframes are used to specify the values for the animating properties at various points during the animation. The keyframes specify the behavior of one cycle of the animation; the animation may iterate one or more times.

Keyframes are specified using a specialized **CSS at-rule**. A **@keyframes** rule consists of the keyword "@keyframes", followed by an identifier giving a name for the animation (which will be referenced using 'animation-name'), followed by a set of style rules (delimited by curly braces).

keyframe syntax:

```
keyframeSelector {


}
```

The keyframe declaration block {…} for a keyframe rule consists of properties and values.

The keyframe selector consists of a comma-separated list of percentage values or the keywords 'from' or 'to'.

Example:

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

The selector is used to specify the percentage along the duration of the animation that the keyframe represents. The keyframe itself is specified by the block of property values declared on the selector. 0% indicates the first moment of the animation sequence, while 100% indicates the final state of the animation.

The keyword 'from' is equivalent to the value '0%'. The keyword 'to' is equivalent to the value '100%'.

@keyframes rule syntax

@keyframes *name* {
    several keyframes
}

After you define @keyframes rule, you can use this rule in an element. In the element, you will include two animation sub properties, **animation-duration** and **animation-name**:

animation-duration: numberofSeconds;
animation-name: *@keyframes rule*;

Browser support note:
Chrome needs the **–webkit-** prefix. Firefox and IE work without any prefixes.

A complete example:

```
<html><head>
<style>
   h1 {
  -webkit-animation-duration: 3s;
  -webkit-animation-name: changeFont;
}
@-webkit-keyframes changeFont {
  from {
    font-size:10px;
  }
  to {
    font-size:80px;
  }
}
```

```
</style>
</head>
<body>
<h1>Hello</h1>
</body>
</html>
```

The name of the @keyframes rule is changeFont and it has two keyframes: one with selector **from** and the other **to**. The rule specifies that the font size grow from 10px to 80px. The rule is applied to <h1> and duration of animation is 3 seconds.

The @keyframes rule selectors from and to specify what should happen at the beginning and at the end. You can specify what should happen at intermediate stages by using percentage selectors.

For example, these keyframes,

```
30%{
   color:green;
 }
 80%{
   color:red;
 }
```

specify that at 30% the color should change to green and at 80% to red.

---

15. Configuring an element for animation

We include the **animation-name** in an element CSS style to associate the element with a @keyframes rule by assigning the property the name of the rule. And the value of the property **animation-duration** determines how long the animation should last. There are other properties we can use in the element's CSS style. Here is a complete list:

animation-delay
Configures the delay between the time the element is loaded and the beginning of the animation sequence.

animation-direction
Configures whether or not the animation should alternate direction on each run through the sequence or reset to the start point and repeat itself.

Possible values:
normal
Initial value. The animation is played in the forward direction.
reverse
The animation cycles are played in the reverse direction of the direction in which they were specified.
alternate

The animation cycles alternate between playing in the forward direction (odd iterations) and the reverse direction (even iterations).
alternate-reverse
The animation cycles alternate between playing in the reverse direction (odd iterations) and the forward direction (even iterations).

animation-duration
Configures the length of time that an animation should take to complete one cycle.

animation-iteration-count
Configures the number of times the animation should repeat; you can specify infinite to repeat the animation indefinitely.
Possible values:
A number
The animation is repeated the specified number of times. Default is 1.

infinite
The animation is repeated forever.

animation-name
Specifies the name of the @keyframes at-rule describing the animation's keyframes.

animation-timing-function
Configures the timing of the animation; that is, how the animation transitions through keyframes, by establishing acceleration curves.

Possible values are:
ease
Gradually increases in speed at the start, animates at full speed, and then gradually decreases in speed at the end.
linear
Consistent speed from start to end.
ease-in
Gradually increases in speed at the start.
ease-out
Gradually decreases in speed at the end.
ease-in-out
Gradually increases in speed at the start and then gradually decreases in speed at the end.

You have to experiment with them to see how each behaves.

---

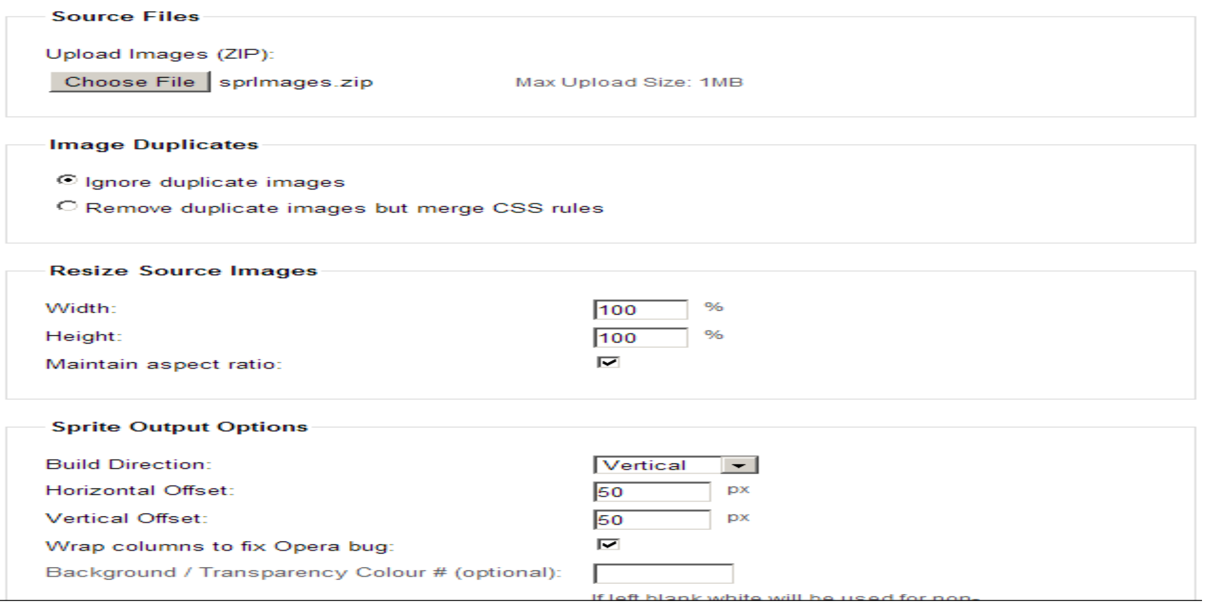
16. CSS sprites

A sprite is single image file, which contains several images to be used in your Web page. If you like to include several images in your page, it takes network resources for each image file to be downloaded by the browser. Instead you can create a sprite file with all your images so that downloading saves time.

Creating a sprite file.
There are tools available, which take your separate image files and create one sprite file. One such tool is here (there are many on the Web, I used this one): http://spritegen.website-performance.org/.

Let us create a sprite with these four files:
First create a folder with your individual images. Zip the folder (you need it to use the sprite generator at http://spritegen.website-performance.org/.
For this example, I have created a folder with these four individual jpg files:

I have zipped the file, sprImages.zip.
Start sprite generator (http://spritegen.website-performance.org/):



Source Files

Upload Images (ZIP):
Choose File sprImages.zip Max Upload Size: 1MB

Image Duplicates

[✓ Ignore duplicate images]
Remove duplicate images but merge CSS rules

Resize Source Images

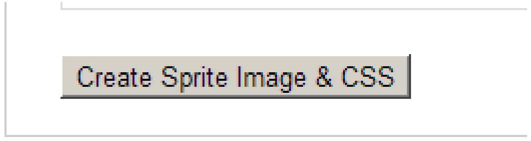
Width:	100	%
Height:	100	%
Maintain aspect ratio:	✓	

Sprite Output Options

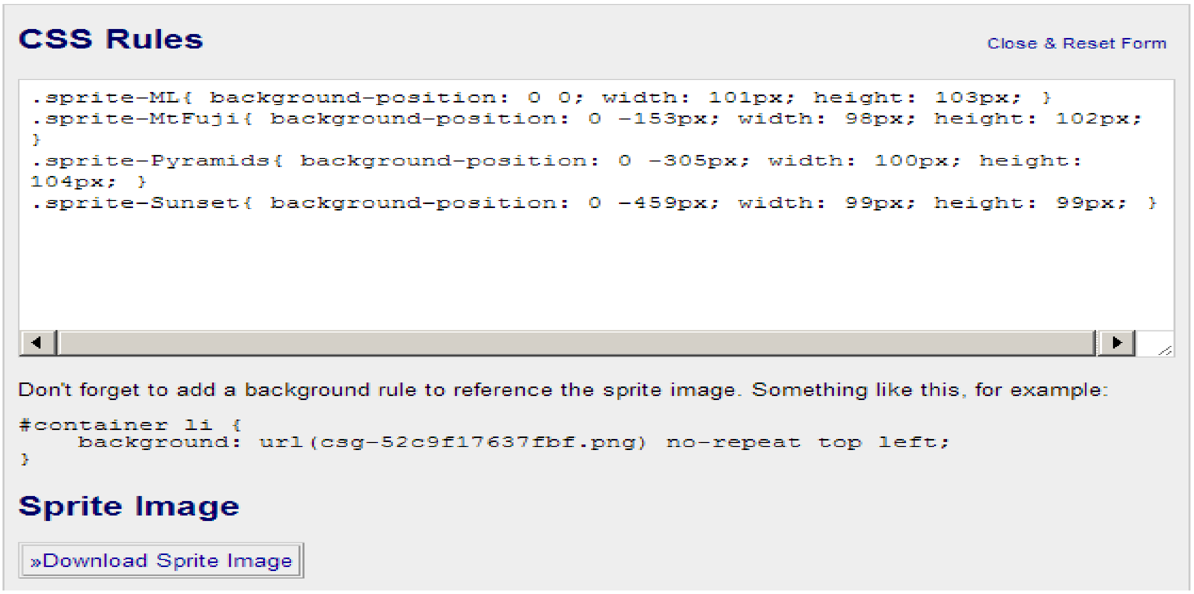
Build Direction:	Vertical	
Horizontal Offset:	50	px
Vertical Offset:	50	px
Wrap columns to fix Opera bug:	✓	
Background / Transparency Colour # (optional):		

If left blank white will be used for non-

Choose File, and leave all defaults as they are. Click,



The result will be this screen:



```
.sprite-ML{ background-position: 0 0; width: 101px; height: 103px; }
.sprite-MtFuji{ background-position: 0 -153px; width: 98px; height: 102px;
}
.sprite-Pyramids{ background-position: 0 -305px; width: 100px; height:
104px; }
.sprite-Sunset{ background-position: 0 -459px; width: 99px; height: 99px; }
```

Don't forget to add a background rule to reference the sprite image. Something like this, for example:

```
#container li {
    background: url(csg-52c9f17637fbf.png) no-repeat top left;
}
```

Click Download Sprite Image button. Also copy the CSS rules text. These CSS rules are important in processing the images.
This is how the sprite image looks:

Let us study the CSS rules:
.sprite-ML{ background-position: 0 0; width: 101px; height: 103px; }

```
.sprite-MtFuji{ background-position: 0 -153px; width: 98px; height: 102px; }
.sprite-Pyramids{ background-position: 0 -305px; width: 100px; height: 104px; }
.sprite-Sunset{ background-position: 0 -459px; width: 99px; height: 99px; }
```

These rules indicate that the image ML starts at 0,0 and its dimensions are 101px and 103px.
Similarly, the image MtFuji starts at 0, -153px and its dimensions are 98px, and 102px. The y-coordinate is negative because y starts at the top, downwards is negative.
Similarly the other two images.

Let us now write HTML and CSS to process the individual images from one sprite image file. Let me show you a part of the code:

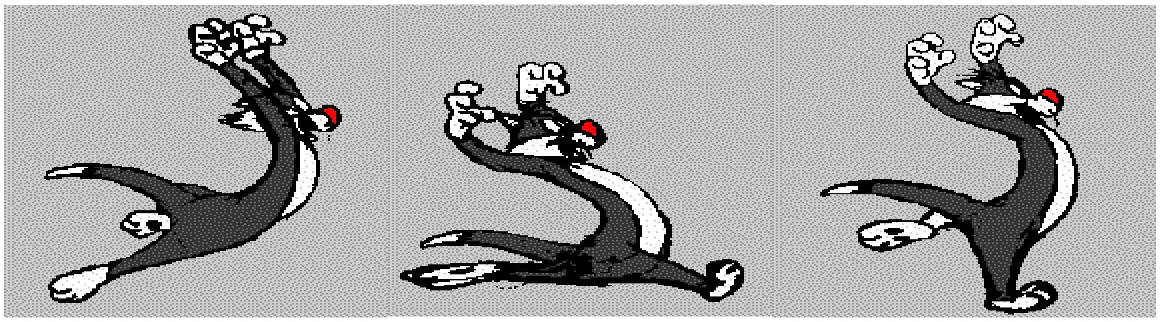
```
#one{
width:100px;
height:100px;
background-color:cyan;
}
#one:hover{
background:url(sprite4Images.png) 0 0 no-repeat;
}

<div id="one"></div>
<div id="two"></div>
```

When the mouse pointer hovers over one, it will display the image at 0,0.

```
#two:hover{
background:url(sprite4Images.png) 0 -153px no-repeat;
}
```

When the mouse pointer hovers over two, it will display the image at 0, -153.
See the complete code here:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter3/spriteExample/spriteExampleHTMLCSS.pdf
Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter3/spriteExample/

Example:
In this example is an animation. First a sequence of individual images are collected:


There are 12 images forming a sequence. These 12 images are made into a sprite png file. The code displays the 12 images in the sprite file one after the other to give the illusion of animation. Study the code, which uses CSS3 animation techniques.

The code is trivial:

```
<html>
<style>
.pics {
 -webkit-animation: showPics 2s steps(12) infinite;
    background: url(catsSprite.png) 0 0 no-repeat ;
height:226px;
width:242px;
margin: 100px auto 0;
}
@-webkit-keyframes showPics {
  0% {background-position: 0 0; }
  100% {background-position: 0  -2712px}
}
</style>
<body>
<h1>Cat is running to catch ...</h1>
<div class="pics"></div>
</body>
<html>
```

Important steps are:
-webkit-animation: showPics 2s steps(12) infinite;
The animation happens in 12 equal steps. To total length of the sprite is -2712px.

Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter3/catRunning/