CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

Mouse over the element below to see a 2D transformation:

2D rotate

In this chapter you will learn about the following CSS property:

- transform

Browser Support

The numbers in the table specify the first browser version that fully supports the property.

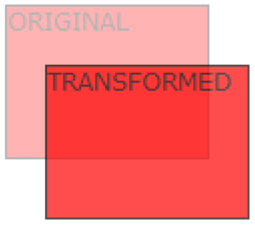
Property	Chrome	Internet Explorer	Firefox	Safari	Opera
transform	36.0	10.0	16.0	9.0	23.0

CSS 2D Transforms Methods

With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()
- matrix()

The translate() Method



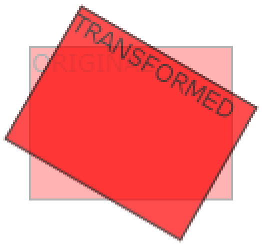
The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position:

Example

```
div {
  transform: translate(50px, 100px);
}
```

The rotate() Method



The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the <div> element clockwise with 20 degrees:

Example

```
div {
 transform: rotate(20deg);
}
```

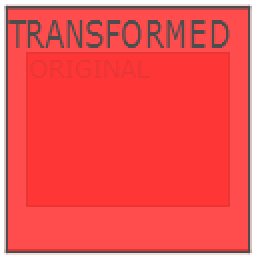
Using negative values will rotate the element counter-clockwise.

The following example rotates the <div> element counter-clockwise with 20 degrees:

Example

```
div {
  transform: rotate(-20deg);
}
```

The scale() Method



The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

The following example increases the <div> element to be two times of its original width, and three times of its original height:

Example

```
div {
  transform: scale(2, 3);
}
```

The scaleX() Method

The scaleX() method increases or decreases the width of an element.

The following example increases the <div> element to be two times of its original width:

Example

```
div {
  transform: scaleX(2);
}
```

The following example decreases the <div> element to be half of its original width:

Example

```
div {
  transform: scaleX(0.5);
}
```

The scaleY() Method

The scaleY() method increases or decreases the height of an element.

The following example increases the <div> element to be three times of its original height:

Example

```
div {
  transform: scaleY(3);
}
```

The following example decreases the <div> element to be half of its original height:

Example

```
div {
  transform: scaleY(0.5);
}
```

The skewX() Method

The skewX() method skews an element along the X-axis by the given angle.

The following example skews the <div> element 20 degrees along the X-axis:

Example

```
div {
  transform: skewX(20deg);
}
```

The skewY() Method

The skewY() method skews an element along the Y-axis by the given angle.

The following example skews the <div> element 20 degrees along the Y-axis:

Example

```
div {
  transform: skewY(20deg);
}
```

The skew() Method

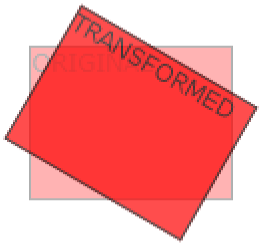
The skew() method skews an element along the X and Y-axis by the given angles.

The following example skews the <div> element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

Example

```
div {
  transform: skew(20deg, 10deg);
}
```

The matrix() Method



The matrix() method combines all the 2D transform methods into one.

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follow: matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

Example

```
div {
  transform: matrix(1, -0.3, 0, 1, 0, 0);
}
```

CSS Transform Properties

The following table lists all the 2D transform properties:

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements

CSS 2D Transform Methods

Function	**Description**
matrix(*n,n,n,n,n,n*)	Defines a 2D transformation, using a matrix of six values

translate(*x*,*y*)	Defines a 2D translation, moving the element along the X- and the Y-axis
translateX(*n*)	Defines a 2D translation, moving the element along the X-axis
translateY(*n*)	Defines a 2D translation, moving the element along the Y-axis
scale(*x*,*y*)	Defines a 2D scale transformation, changing the elements width and height
scaleX(*n*)	Defines a 2D scale transformation, changing the element's width
scaleY(*n*)	Defines a 2D scale transformation, changing the element's height
rotate(*angle*)	Defines a 2D rotation, the angle is specified in the parameter
skew(*x-angle*,*y-angle*)	Defines a 2D skew transformation along the X- and the Y-axis
skewX(*angle*)	Defines a 2D skew transformation along the X-axis

skewY(*angle*)	Defines a 2D skew transformation along the Y-axis

CSS 3D Transforms

CSS also supports 3D transformations.

Mouse over the elements below to see the difference between a 2D and a 3D transformation:

2D rotate

3D rotate

In this chapter you will learn about the following CSS property:

- transform

Browser Support

The numbers in the table specify the first browser version that fully supports the property.

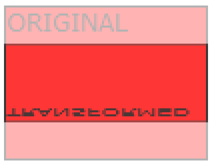
Property	Chrome	Internet Explorer	Firefox	Safari	Opera
transform	36.0	10.0	16.0	9.0	23.0

CSS 3D Transforms Methods

With the CSS transform property you can use the following 3D transformation methods:

- rotateX()
- rotateY()
- rotateZ()

The rotateX() Method

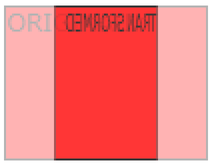


The rotateX() method rotates an element around its X-axis at a given degree:

Example

```
#myDiv {
  transform: rotateX(150deg);
}
```

The rotateY() Method



The rotateY() method rotates an element around its Y-axis at a given degree:

Example

```
#myDiv {
  transform: rotateY(130deg);
}
```

The rotateZ() Method

The rotateZ() method rotates an element around its Z-axis at a given degree:

Example

```
#myDiv {
  transform: rotateZ(90deg);
}
```

CSS 3D Transform Methods

Function	**Description**
matrix3d (*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*)	Defines a 3D transformation, using a 4x4 matrix of 16 values
translate3d(*x*,*y*,*z*)	Defines a 3D translation

translateX(*x*)	Defines a 3D translation, using only the value for the X-axis
translateY(*y*)	Defines a 3D translation, using only the value for the Y-axis
translateZ(*z*)	Defines a 3D translation, using only the value for the Z-axis
scale3d(*x*,*y*,*z*)	Defines a 3D scale transformation
scaleX(*x*)	Defines a 3D scale transformation by giving a value for the X-axis
scaleY(*y*)	Defines a 3D scale transformation by giving a value for the Y-axis
scaleZ(*z*)	Defines a 3D scale transformation by giving a value for the Z-axis
rotate3d(*x*,*y*,*z*,*angle*)	Defines a 3D rotation
rotateX(*angle*)	Defines a 3D rotation along the X-axis

rotateY(*angle*)	Defines a 3D rotation along the Y-axis
rotateZ(*angle*)	Defines a 3D rotation along the Z-axis
perspective(*n*)	Defines a perspective view for a 3D transformed element

CSS Animations

What are CSS Animations?

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

CSS allows animation of HTML elements without using JavaScript or Flash!

Properties:

- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

Property	**Chrome**	**Internet Explorer**	**Firefox**	**Safari**	**Opera**
@keyframes	43.0	10.0	16.0	9.0	30.0

animation-name	43.0	10.0	16.0	9.0	30.0
animation-duration	43.0	10.0	16.0	9.0	30.0
animation-delay	43.0	10.0	16.0	9.0	30.0
animation-iteration-count	43.0	10.0	16.0	9.0	30.0
animation-direction	43.0	10.0	16.0	9.0	30.0
animation-timing-function	43.0	10.0	16.0	9.0	30.0
animation-fill-mode	43.0	10.0	16.0	9.0	30.0
animation	43.0	10.0	16.0	9.0	30.0

Browser Support for Animations

The numbers in the table specify the first browser version that fully supports the property.

The @keyframes Rule

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

```
/* The animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

Set How Many Times an Animation Should Run

The animation-iteration-count property specifies the number of times an animation should run.

The following example will run the animation 3 times before it stops:

Example

```
<!DOCTYPE html>

<html>

<head>

<style>

div {

  width: 100px;

  height: 100px;

  background-color: red;

  position: relative;

  animation-name: example;

  animation-duration: 4s;

  animation-iteration-count: 3;

}


@keyframes example {

  0%   {background-color:red; left:0px; top:0px;}

  25%  {background-color:yellow; left:200px; top:0px;}

  50%  {background-color:blue; left:200px; top:200px;}

  75%  {background-color:green; left:0px; top:200px;}

  100% {background-color:red; left:0px; top:0px;}

}
```

```
</style>

</head>

<body>


<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>


<div></div>


</body>

</html>
```

The following example uses the value "infinite" to make the animation continue for ever:

```
<!DOCTYPE html>

<html>

<head>

<style>

div {

  width: 100px;

  height: 100px;

  background-color: red;

  position: relative;

  animation-name: example;

  animation-duration: 4s;

  animation-iteration-count: infinite;

}


@keyframes example {

  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}

  50%  {background-color:blue; left:200px; top:200px;}

  75%  {background-color:green; left:0px; top:200px;}

  100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>


<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>


<div></div>


</body>

</html>
```

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	**Description**
@keyframes	Specifies the animation code
animation	A shorthand property for setting all the animation properties
animation-delay	Specifies a delay for the start of an animation

animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles
animation-duration	Specifies how long time an animation should take to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies the name of the @keyframes animation
animation-play-state	Specifies whether the animation is running or paused
animation-timing-function	Specifies the speed curve of the animation

Animation Delay

The animation-delay property specifies a delay for the start of an animation.

The animation-delay value is defined in seconds (s) or milliseconds (ms).

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation: mymove 5s infinite;
  animation-delay: 2s;
}

@keyframes mymove {
  from {left: 0px;}
  to {left: 200px;}
}
</style>
</head>
<body>

<h1>The animation-delay Property</h1>
<p>Start the animation after 2 seconds:</p>
<div></div>
</body>
</html>
```

Animation Direction

The animation-direction property defines whether an animation should be played forwards, backwards or in alternate cycles.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation: example 5s 2;
  animation-direction: alternate;
}
@keyframes example {
  0%   {background: red; left: 0px; top: 0px;}
  25%  {background: yellow; left: 200px; top: 0px;}
  50%  {background: blue; left: 200px; top: 200px;}
  75%  {background: green; left: 0px; top: 200px;}
  100% {background: red; left: 0px; top: 0px;}
}
</style>
</head>
<body>
<h1>animation-direction: alternate</h1>
<p>Play the animation forwards first, then backwards:</p>
<div></div>
</body>
</html>
```