CSS3

TRANSFORM

The CSS3 **transform property** allows us to translate, rotate, scale or skew any element on the page in 2D or 3D.

We can **manipulate elements** using a range of transform functions including:

```
p { transform: translate(x,y); }
p { transform: translateX(x); }
p { transform: translateY(y); }

/* not covered in this presentation */
p { transform: translateZ(z); }
p { transform: translate3d(x,y,z); }
```

```
p { transform: rotate(angle); }
p { transform: rotateX(angle); }
p { transform: rotateY(angle); }

/* not covered in this presentation */
p { transform: rotateZ(angle); }
p { transform: rotate3d(x,y,z,angle); }
```

```
p { transform: scale(x,y); }
p { transform: scaleX(x); }
p { transform: scaleY(y); }

/* not covered in this presentation */
p { transform: scaleZ(z); }
p { transform: scale3d(x,y,z); }
```

```
p { transform: skew(x-angle,y-angle); }
p { transform: skewX(angle); }
p { transform: skewY(angle); }
```

```
/* not covered in this presentation */
p { transform: perspective(n); }
```

```
/* not covered in this presentation */
p { transform: matrix(n,n,n,n,n,n); }
p { transform:
matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n); }
```

transform

The **transform property** allows for one or more space-separated functions.

```
p
{
    transform:
        translate(x,y)
        rotate(angle)
        scale(x,y)
        skew(x-angle,y-angle);
}
```

The **initial value** for transform is “none”.

```
p { transform: none; }
```

The **initial value** sets this property to its default value.

```
p { transform: initial; }
```

The **inherit value** inherits this property from its parent element.

```
p { transform: inherit; }
```

translate()

The **translate() function** allows us to move elements away from their current position.

The **value** is written with translate, followed by ‘(‘ followed by an X co-ordinate, followed by a comma, followed by a Y co-ordinate, followed by ‘)’.

```
p { transform: translate(X, Y); }

p { transform: translate(25px, 5px); }
```

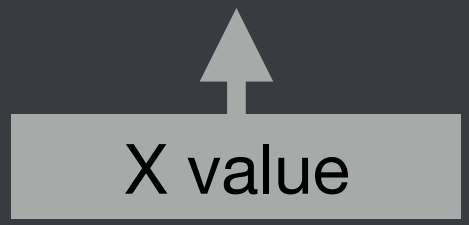
If no second value is provided, the y value is **defined as “0”**.

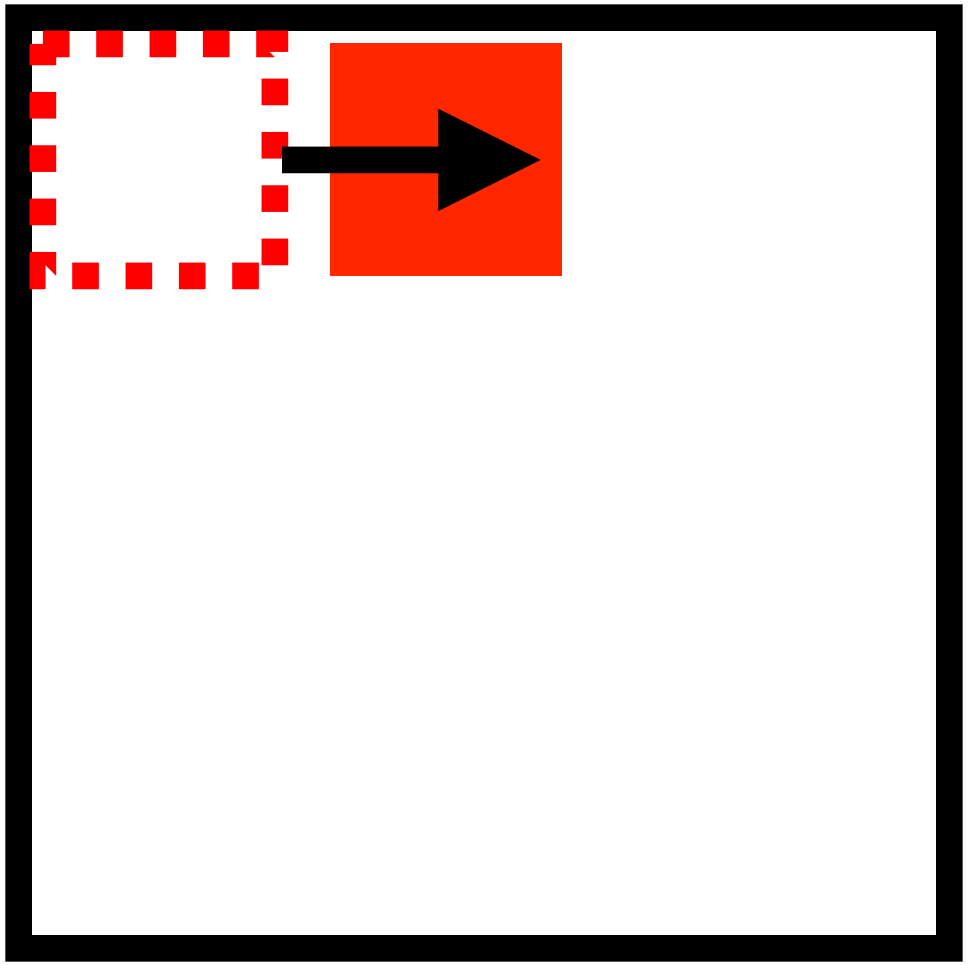
```
p { transform: translate(25px); }

p { transform: translate(25px[, 0]); }
```

A positive **x value** will move the element to the right of its current position.

```
p { transform: translate(25px, 5px); }
```

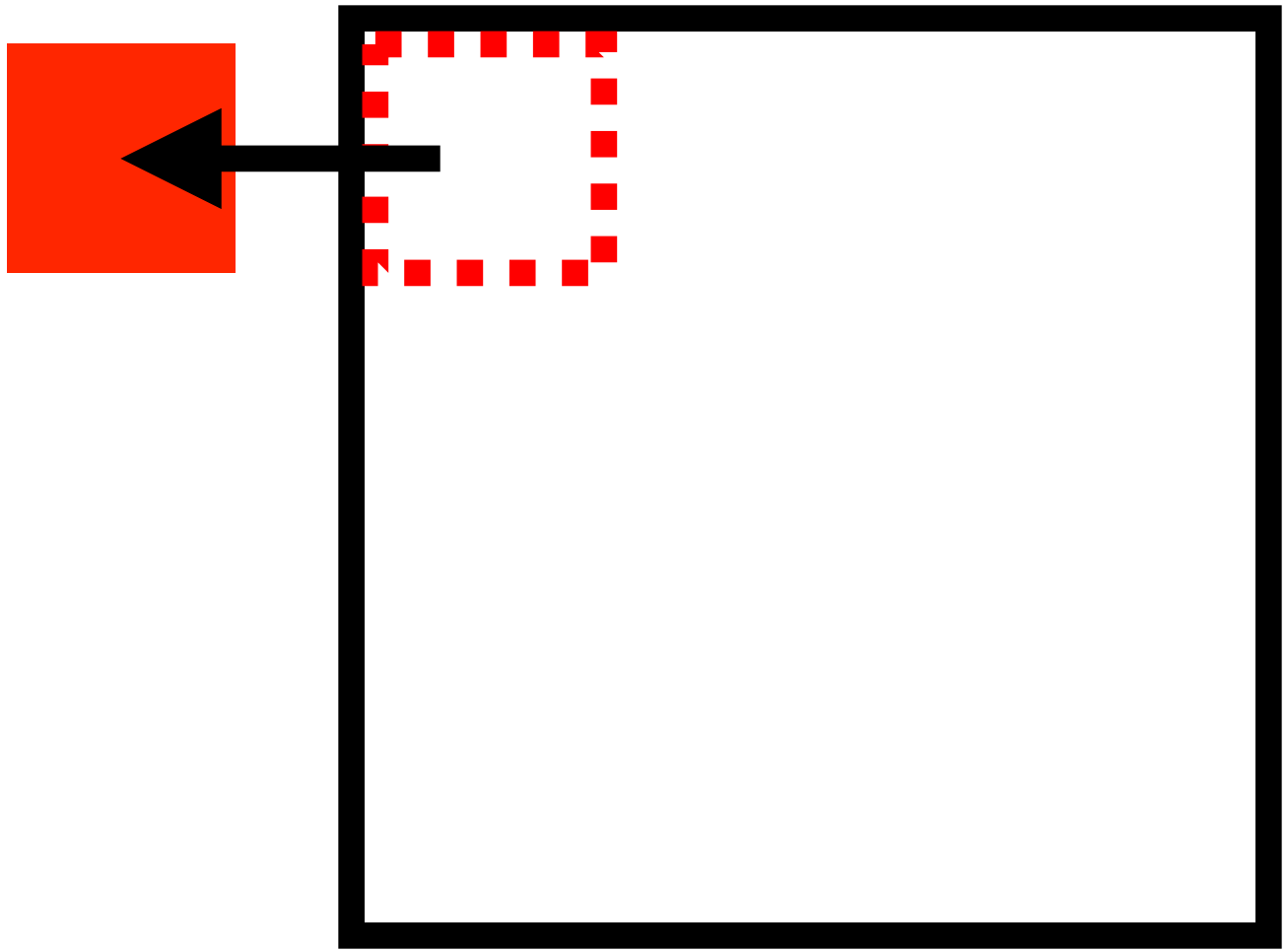




A negative **x value** will move the element to the left of its current position.

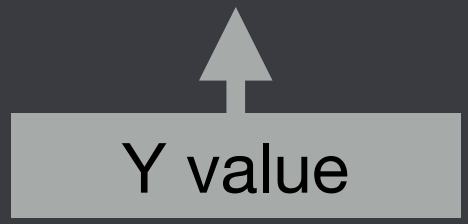
```
p { transform: translate(-25px, 5px); }
```

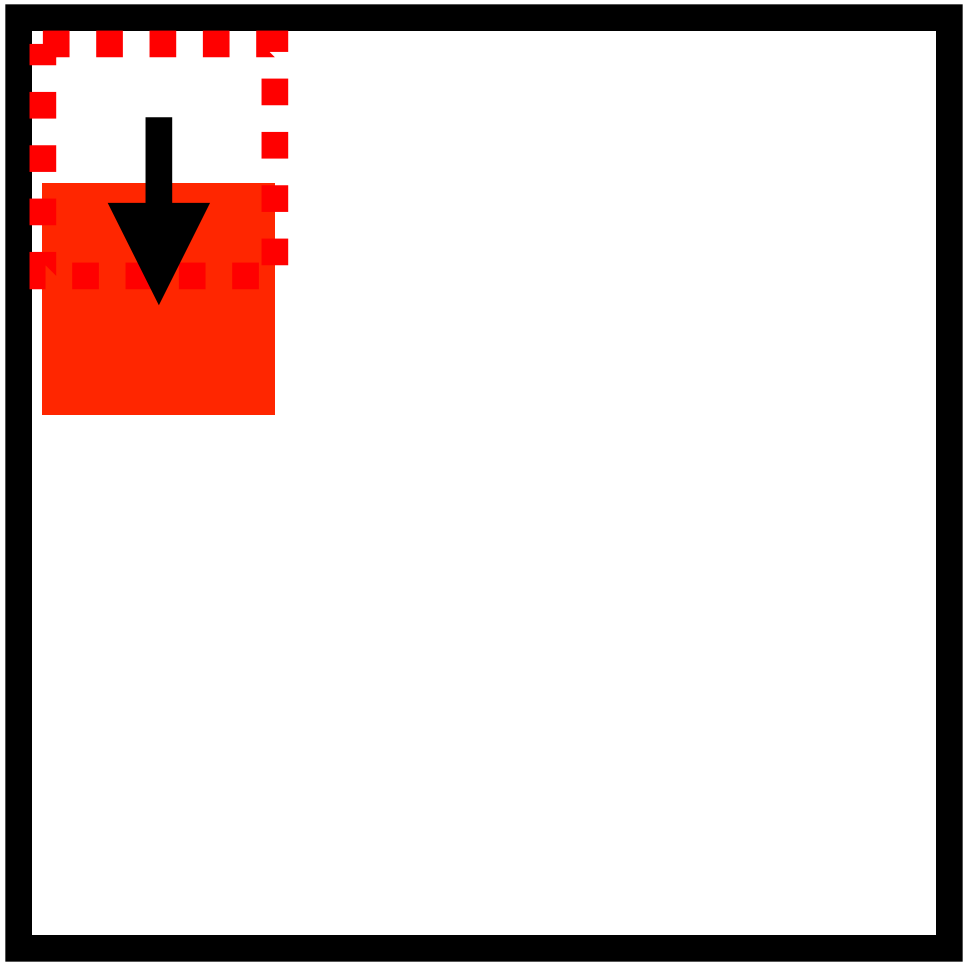
Negative X value



A positive **y value** will move the element down from its current position.

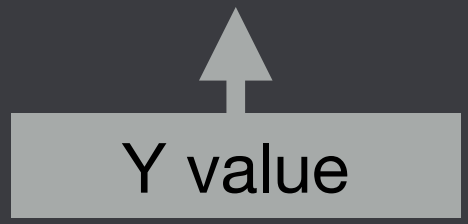
```
p { transform: translate(25px, 5px); }
```

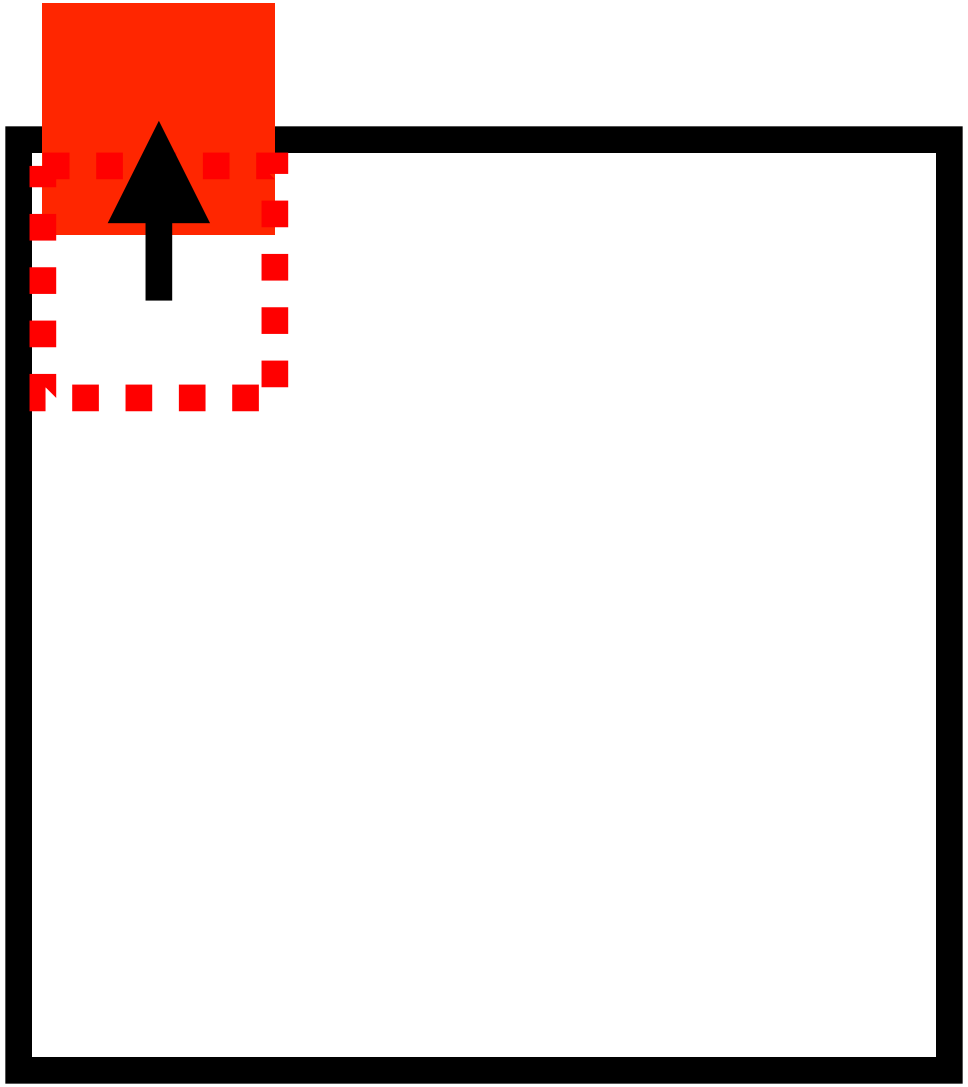




A negative **y value** will move the element up from its current position.

```
p { transform: translate(25px, -5px); }
```





The translate function **does not work on inline elements**. These elements need to be converted to inline-block or block in order for any translate functions to take effect.

Content that comes after translated elements will **completely ignore any new position of the translated element**, and keep a space where the element was - similar to position: relative;

translateX()

If you want to move an element against the x-axis only, you can use **translateX()** which takes a single value.

```
p { transform: translateX(25px); }
```

translateY()

If you want to move an element against the Y-axis only, you can use **translateY()** which takes a single value.

```
p { transform: translateY(25px); }
```

rotate()

The **rotate() function** rotates an element around the point of origin by a specified angle value.

The **value** is written with rotate, followed by ‘(‘ followed by an angle value followed by ‘)’.

Generally, angles are **declared in degrees**, with positive degrees moving clockwise and negative degrees moving counter-clockwise.

```
p { transform: rotate(5deg); }
p { transform: rotate(-5deg); }
```

However, values can also be declared in **grads, radians, or turns**.

deg (degrees)

Each unit is equal to 1/360th of a circle

```
p { transform: rotate(5deg); }
```

grad (grads)

Each unit is equal to 1/400th of a circle

```
p { transform: rotate(3grad); }
```

rad (radians)

2pi radians equal one circle. One radian is equal to about 57.2958 degrees. 2pi radians would be 2 x 57.2958 x 3.1416 = 360 degrees.

```
p { transform: rotate(2rad); }
```

turn (turns)

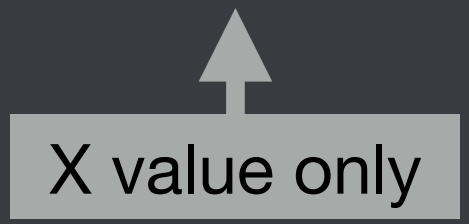
Each unit is equal to one full circle, so the unit “1” would appear exactly the same. However, this value is useful when combined with animate.

```
p { transform: rotate(3turn); }
```

rotateX()

If you want to rotate an element against the x-axis only, you can use **rotateX()** which takes a single value.

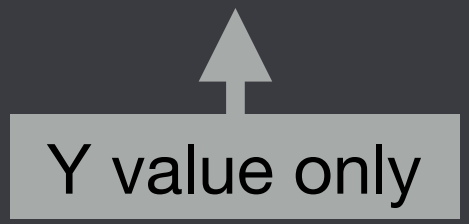
```
p { transform: rotateX(5deg); }
```



rotateY()

If you want to rotate an element against the Y-axis only, you can use **rotateY()** which takes a single value.

```
p { transform: rotateY(5deg); }
```



scale()

The **scale() function** allows us to scale an element up or down in size form the element's original size.

The **value** is written with scale, followed by ‘(‘ followed by an x and y comma-separated co-ordinates followed by ‘)’.

```
p { transform: scale(1.5, 2); }
```

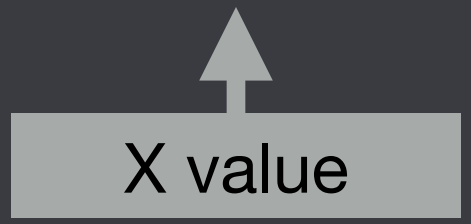
If no second value is provided, the y value is defined as **“0”**.

```
p { transform: scale(1.5); }

p { transform: scale(1.5[, 0]); }
```

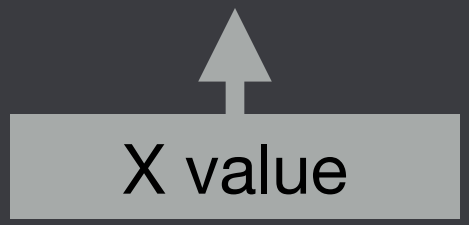
The **x value** scales the element on the X axis.

```
p { transform: scale(1.5, 2); }
```



The **y value** scales the element on the y axis.

```
p { transform: scale(1.5, 2); }
```



scaleX()

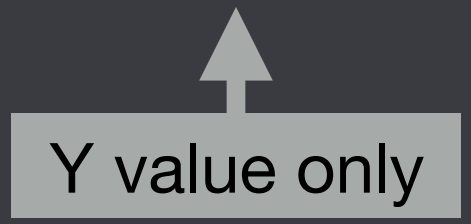
If you want to scale an element against the x-axis only, you can use **scaleX()** which takes a single value.

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

scaleY()

If you want to scale an element against the Y-axis only, you can use **scaleY()** which takes a single value.

```
p { transform: scaleY(2); }
```



skew()

The **skew() function** defines how an element will be skewed as an angle along the x and y axis.

The **value** is written with skew, followed by ‘(‘ followed by an X angle value, followed by a comma, followed by a Y angle value, followed by ‘)’.

```
p { transform: skew(20deg, 50deg); }
```

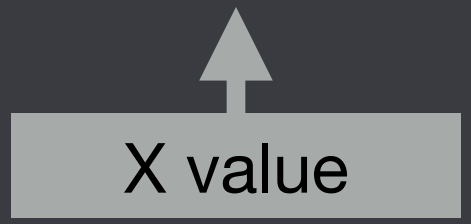
If no second value is provided, the y value is **defined as “0”**.

```
p { transform: skew(20deg); }

p { transform: skew(25px[, 0]); }
```

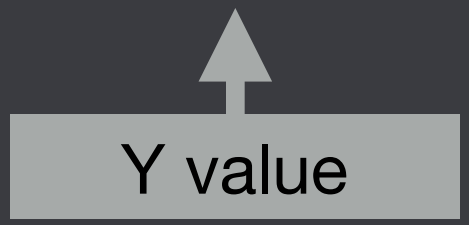
The **x value** skews the element away from the left.

```
p { transform: skew(20deg, 50deg); }
```



The **y value** skews the element away from the top.

```
p { transform: skew(20deg, 50deg); }
```



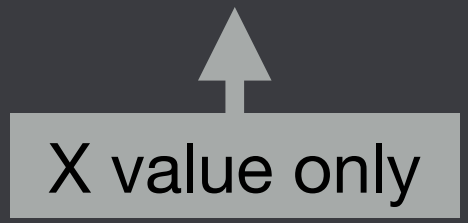
The skew values can be specified using **degrees, radians or gradians**.

```
p { transform: skew(5deg, 1deg); }
p { transform: skew(3grad, 2grad); }
p { transform: skew(2rad, 1rad); }
```

skewX()

If you want to skew an element against the x-axis only, you can use **skewX()** which takes a single value.

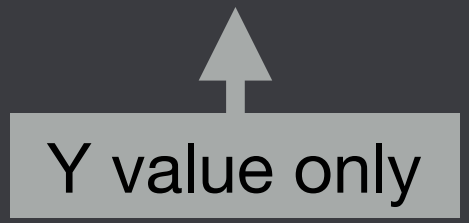
```
p { transform: skewX(20deg); }
```



skewY()

If you want to skew an element against the Y-axis only, you can use **skewY()** which takes a single value.

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



transform-origin

The **transform-origin property** lets us define the origin of transforms for an element.

```
p { transform-origin: 0 0; }
```

The **initial value** is 50% 50% - the center of the element

```
p { transform-origin: 50% 50%; }
```

Length, percentage and keyword values are allowed. Negative percentage and length values are allowed.

```
/* length values */
p { transform-origin: 10px -20px; }
p { transform-origin: -2em 20em; }

/* percentage values */
p { transform-origin: 5% -20%; }

/* keyword values */
p { transform-origin: left top; }
p { transform-origin: right center; }
```

If only one value is specified, the second value is assumed to be **“center”**.

```
p { transform-origin: 10px;}

p { transform-origin: 10px [center];}
```

Browser support

CSS3 2D Transforms - WD

Global 91.39%

unprefixed: 71.57%

Method of transforming an element including rotating, scaling, etc. Includes support for `transform` as well as `transform-origin` properties.

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
								4.1	
8			43					4.3	
9		40	44					4.4	
10		41	45	8		8.4		4.4.4	
11	12	42	46	9	32	9.1	8	44	46
	13	43	47		33				
		44	48		34				
		45	49						

CSS3 3D Transforms - WD

Global	81.81%	+ 8.06% =	89.87%
unprefixed:	63.33%	+ 8.06% =	71.39%

Method of transforming an element in the third dimension using the `transform` property. Includes support for the `perspective` property to set the perspective in z-space and the `backface-visibility` property to toggle display of the reverse side of a 3D-transformed element.

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
								4.1	
8			43					4.3	
9		40	44					4.4	
10 [1]		41	45	8		8.4		4.4.4	
11 [1]	12	42	46	9 [2]	32	9.1	8	44	46
	13	43	47		33				
		44	48		34				
		45	49						

Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley