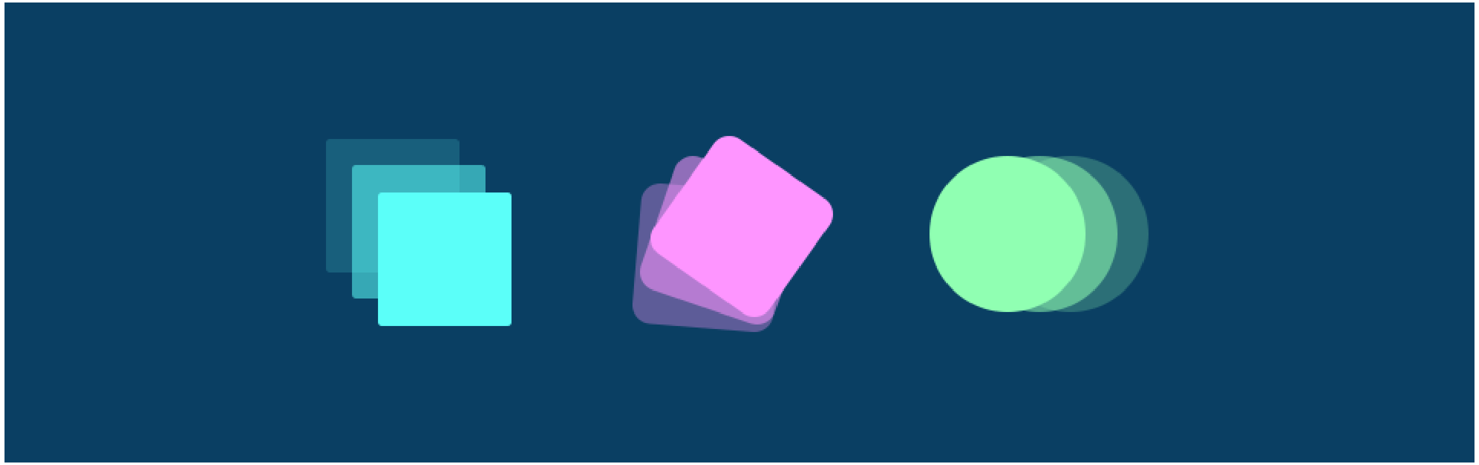
CSS Transform, Transition and Animation

CLARUSWAY
WAY TO REINVENT YOURSELF

Table of Contents



Did you finish pre-class material?

1

CSS 2D Transforms

CLARUSWAY
WAY TO REINVENT YOURSELF

CSS 2D TRANSFORMS

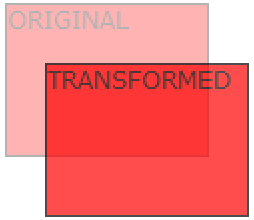
CSS transforms allow you to move, rotate, scale, and skew elements.

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).

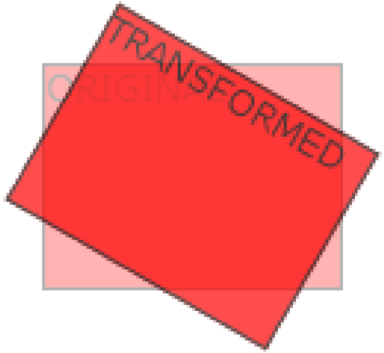


```
div {
  transform: translate(50px, 100px);
}
```

The rotate() Method

The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.Using negative values will rotate the element counter-clockwise.

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



CLARUSWAY
WAY TO REINVENT YOURSELF

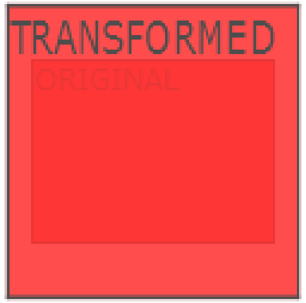
The scale() Method

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

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

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

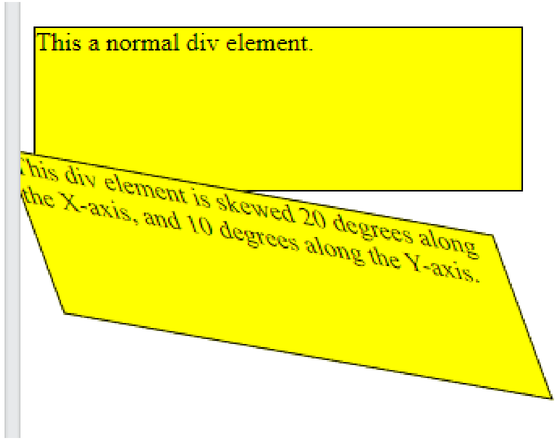
```
div {
   transform: scale(2, 3);
   transform: scaleX(0.5);
   transform: scaleY(3);
}
```



The skew() Method

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

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



This a normal div element.

This div element is skewed 20 degrees along the X-axis, and 10 degrees along the Y-axis.

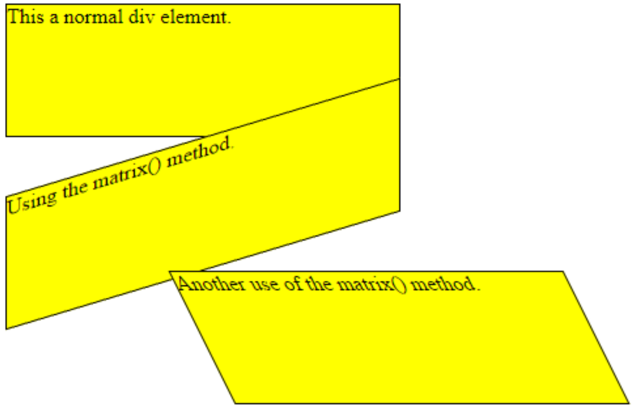
CLARUSWAY
WAY TO REINVENT YOURSELF

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())

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


This a normal div element.
Using the matrix() method.
Another use of the matrix() method.

2

CSS 3D Transforms

CLARUSWAY
WAY TO REINVENT YOURSELF

CSS 3D Transforms

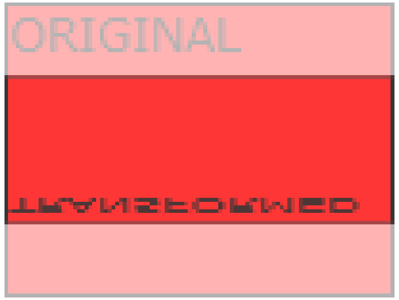
CSS also supports 3D transformations.

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

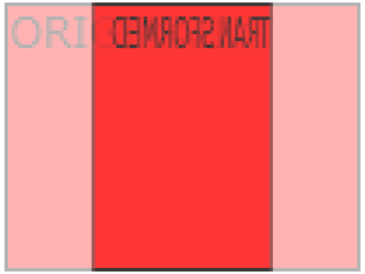


```
#myDiv {
  transform: rotateX(150deg);
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

The rotateY() Method

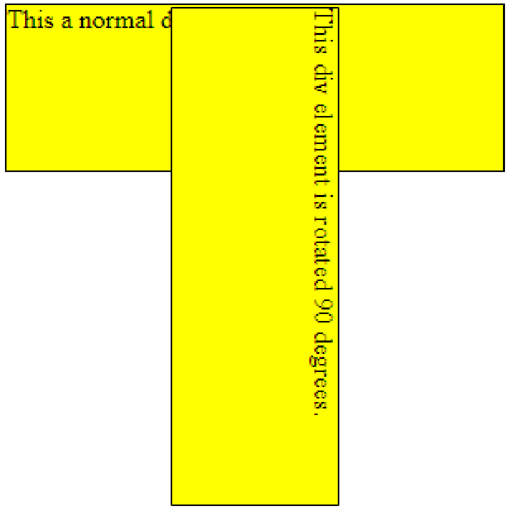
The rotateY() method rotates an element around its Y-axis at a given degree



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

The rotateZ() Method

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



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

3 CSS Transitions

CSS Transitions

CSS transitions allows you to change property values smoothly, over a given duration.

In this chapter you will learn about the following properties: ====>>>

- `transition`
- `transition-delay`
- `transition-duration`
- `transition-property`
- `transition-timing-function`

CLARUSWAY
WAY TO REINVENT YOURSELF

How to Use CSS Transitions?

To create a transition effect, you must specify two things:

1-)The CSS property you want to add an effect to

2-)The duration of the effect

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

```css
div {
  transition: width 2s, height 4s;
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Specify the Speed Curve of the Transition

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

- **ease -** specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- **linear -** specifies a transition effect with the same speed from start to end
- **ease-in -** specifies a transition effect with a slow start
- **ease-out -** specifies a transition effect with a slow end
- **ease-in-out -** specifies a transition effect with a slow start and end
- **cubic-bezier(n,n,n,n) -** lets you define your own values in a cubic-bezier function

Example

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

Delay the Transition Effect

The transition-delay property specifies a delay (in seconds) for the transition effect.

The following example has a 1 second delay before starting:

```
div {
  transition-delay: 1s;
}
```

More Transition Example

```
div {
   transition-property: width;
   transition-duration: 2s;
   transition-timing-function: linear;
   transition-delay: 1s;
}
```

3

CSS Animations

CLARUSWAY
WAY TO REINVENT YOURSELF

CSS Transitions

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

In this chapter you will learn about the following properties: ====>>>

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

CLARUSWAY
WAY TO REINVENT YOURSELF

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 as 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.

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.

Example

```
/* 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;
}
```

Delay an Animation

The animation-delay property specifies a delay for the start of an animation.Negative values are also allowed.

The following example has a 2 seconds delay before starting the animation:

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-delay: 2s;
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Run Animation in Reverse Direction or Alternate Cycles

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

The animation-direction property can have the following values:

- **normal -** The animation is played as normal (forwards). This is default
- **reverse -** The animation is played in reverse direction (backwards)
- **alternate -** The animation is played forwards first, then backwards
- **alternate-reverse -** The animation is played backwards first, then forwards

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-direction: reverse;
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Specify the Speed Curve of the Animation

The animation-timing-function property specifies the speed curve of the animation.

The animation-timing-function property can have the following values:

- **ease -** Specifies an animation with a slow start, then fast, then end slowly (this is default)
- **linear -** Specifies an animation with the same speed from start to end
- **ease-in -** Specifies an animation with a slow start
- **ease-out -** Specifies an animation with a slow end
- **ease-in-out -** Specifies an animation with a slow start and end
- **cubic-bezier(n,n,n,n) -** Lets you define your own values in a cubic-bezier function

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

CLARUSWAY
WAY TO REINVENT YOURSELF

Specify the fill-mode For an Animation

CSS animations do not affect an element before the first keyframe is played or after the last keyframe is played. The animation-fill-mode property can override this behavior.The animation-fill-mode property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both).

The animation-fill-mode property can have the following values:

- **none -** Default value. Animation will not apply any styles to the element before or after it is executing
- **forwards -** The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
- **backwards -** The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
- **both -** The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions

```
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation-name: example;
  animation-duration: 3s;
  animation-fill-mode: forwards;
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Animation Shorthand Property

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

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

THANKS!

Any questions?