## **CSS Animations**

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

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

## **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.

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":

# CSS Animation

**CSS Animation property** is used to create animation on the webpage. It can be used as a replacement of animation created by Flash and [JavaScript](https://javatpoint.com/javascript-tutorial).

## **CSS3 @keyframes Rule**

The animation is created in the @keyframe rule. It is used to control the intermediate steps in a [CSS](https://javatpoint.com/css-tutorial) animation sequence.

## **What animation does**

An animation makes an element change gradually from one style to another. You can add as many as properties you want to add. You can also specify the changes in percentage.0% specify the start of the animation and 100% specify its completion.

## **How CSS animation works**

When the animation is created in the [@keyframe rule](https://www.javatpoint.com/css-keyframes-rule), it must be bound with selector; otherwise the animation will have no effect.

The animation could be bound to the selector by specifying at least these two properties:

* The name of the animation
* The duration of the animation

## **CSS animation properties**

|  |  |
| --- | --- |
| **Property** | **Description** |
| @keyframes | It is used to specify the animation. |
| Animation | This is a shorthand property, used for setting all the properties, except the animation-play-state and the animation-fill- mode property. |
| animation-delay | It specifies when the animation will start. |
| animation-direction | It specifies if or not the animation should play in reserve on alternate cycle. |
| animation-duration | It specifies the time duration taken by the animation to complete one cycle. |
| animation-fill-mode | it specifies the static style of the element. (when the animation is not playing) |
| animation-iteration-count | It specifies the number of times the animation should be played. |
| animation-play-state | It specifies if the animation is running or paused. |
| animation-name | It specifies the name of @keyframes animation. |
| animation-timing-function | It specifies the speed curve of the animation. |

## **CSS animation example: changing background color**

Let's see a simple CSS animation example that changes background color of rectangle from RED to BLACK.

## **CSS animation example: changing background color**

Let's see a simple CSS animation example that changes background color of rectangle from RED to BLACK.

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<style>**
5. div {
6. width: 100px;
7. height: 100px;
8. background: red;
9. -webkit-animation: myfirst 6s; /\* Chrome, Safari, Opera \*/
10. animation: myfirst 5s;
11. }
12. /\* Chrome, Safari, Opera \*/
13. @-webkit-keyframes myfirst {
14. from {background: red;}
15. to {background: green;}
16. }
17. /\* Standard syntax \*/
18. @keyframes myfirst {
19. from {background: red;}
20. to {background: green;}
21. }
22. **</style>**
23. **</head>**
24. **<body>**
25. **<p><b>**Note:**</b>** The IE 9 and earlier versions don't support CSS3 animation property. **</p>**
26. **<div></div>**
27. **</body>**
28. **</html>**

## **CSS animation example: Moving Rectangle**

Let's take another example to display animation with percentage value.

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<style>**
5. div {
6. width: 100px;
7. height: 100px;
8. background: red;
9. position: relative;
10. -webkit-animation: myfirst 5s; /\* Chrome, Safari, Opera \*/
11. animation: myfirst 5s;
12. }
13. /\* Chrome, Safari, Opera \*/
14. @-webkit-keyframes myfirst {
15. 0%   {background:red; left:0px; top:0px;}
16. 25%  {background:yellow; left:300px; top:0px;}
17. 50%  {background:blue; left:200px; top:300px;}
18. 75%  {background:green; left:0px; top:200px;}
19. 100% {background:red; left:0px; top:0px;}
20. }
21. /\* Standard syntax \*/
22. @keyframes myfirst {
23. 0%   {background:red; left:0px; top:0px;}
24. 25%  {background:yellow; left:300px; top:0px;}
25. 50%  {background:blue; left:300px; top:200px;}
26. 75%  {background:green; left:0px; top:200px;}
27. 100% {background:red; left:0px; top:0px;}
28. }
29. **</style>**
30. **</head>**
31. **<body>**
32. **<p><b>**Note:**</b>** The Internet Explorer 9 and its earlier versions don't support this example.**</p>**
33. **<div></div>**
34. **</body>**
35. **</html>**

# CSS @keyframes rule

The CSS @keyframe specifies the animation rule that defines the CSS properties for the elements at each state with a timeline.

We can create complex animation properties by using the **@keyframe.** An animation is created with the changeable CSS styles that can be repeated indefinitely or a finite number of times. A simple animation can just have two keyframes, while the complex animation has several keyframes.

### **Syntax**

1. @keyframes animation-name {keyframes-selector {css-styles;}}

To use keyframes, we need to create a **@keyframes** rule along with the **animation-name** property for matching an animation with its keyframe declaration.

It accepts three values. Let's discuss each of them in detail.

### **Property values**

**animation-name:** It is the required value that defines the name of the animation.

**keyframes-selector:** It specifies the percentage along with the animation when the keyframe occurs. Its value lies between 0% to 100%, from (same as 0%), to (same as 100%). Keyframe percentages can be written in any order because they will be handled in the order they occur.

**css-styles:** It defines one or more than one CSS style properties.

If the keyframe rule does not define the start and end states of animation, then the browsers will use the existing styles of the element. The properties get ignored that cannot be animated in keyframe rules.

## **Keyframes timing function**

For controlling the animation rate, we can use the following values.

**linear:** It means that the transition rate will be constant from start to end.

**ease:** It means that the animation starts slowly, and then after a time period, the speed increases, and before end speed will again slow down.

**ease-in:** It is similar to ease, but the animation ends quickly.

**ease-out:** It is also similar to ease, but the animation starts fast.

Let us try to understand CSS @keyframe rule with some illustrations.

### **Example**

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<style>**
5. div
6. {
7. width:200px;
8. height:200px;
9. animation:demo 5s ease-in infinite, trans 5s ease-in-out infinite;
10. border-radius:40px;
11. }
13. @keyframes demo
14. {
15. 0% {background:red;}
16. 50% {background:yellow; width:100px; height:100px;}
17. 100% {background:green; width:300px; height:300px;}
18. }
19. @keyframes trans
20. {
21. 0% {transform:translate(0px) scale(1.4) rotate(80deg);}
22. 50% {transform:translate(250px) scale(1.2) rotate(40deg);}
23. 100% {transform:translate(350px) scale(1) rotate(0deg);}
24. }
26. **</style>**
27. **</head>**
28. **<body>**
30. **<div></div>**
31. **<p>**After the completion of the Animation, the element retracts to its original State **</p>**
33. **</body>**
34. **</html>**

### **Example2**

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<style>**
5. h1 {
6. color: black;
7. text-align: center;
8. }
9. div {
10. position: relative;
11. animation: jtp 7s infinite;
12. }
14. @keyframes jtp {
15. 0% {
16. top: 500px;
17. width: 0px;
18. font-size:10px;
19. transform: translate(0px) scale(1.4) rotate(80deg);
20. }
21. 25% {
22. top: 400px;
23. background: yellow;
24. font-size:20px;
25. width: 50px;
26. transform: translate(100px) scale(1.3) rotate(60deg);
27. }
28. 50% {
29. top: 300px;
30. background: orange;
31. font-size:30px;
32. width: 150px;
33. transform: translate(200px) scale(1.2) rotate(40deg);
34. }
35. 75% {
36. top: 200px;
37. background: pink;
38. width: 250px;
39. font-size:40px;
40. transform: translate(300px) scale(1.1) rotate(20deg);
41. }
42. 100% {
43. top: 100px;
44. background: red;
45. font-size:50px;
46. width: 500px;
47. transform: translate(400px) scale(1) rotate(0deg);
48. }
49. }
50. **</style>**
51. **</head>**
53. **<body>**
54. **<div>**
55. **<h1>**javaTpoint**</h1>**
56. **</div>**
57. **</body>**
59. **</html>**

## **Points to remember**

Some of the important points about this rule are given as follows:

* We can use **from** keyword instead of using **0%**.
* We can use **to** keyword instead of using **100%**.
* Even if we are using **from** and **to** keywords, any values between them will still be declared in **%.**
* For the valid animation, the starting and ending time must be declared.
* Those declarations get ignored that involves the **!important**

## **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

The following example will run the animation in reverse direction (backwards):

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

animation-direction: reverse;

}

@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>

<h1>CSS Animation</h1>

<p>The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles. The following example will run the animation in reverse direction (backwards):</p>

<div></div>

</body>

</html>

Example2

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background-color: red;

position: relative;

animation-name: example;

animation-duration: 4s;

animation-iteration-count: 2;

animation-direction: alternate;

}

@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>

<h1>CSS Animation</h1>

<p>The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles. The following example uses the value "alternate" to make the animation run forwards first, then backwards:</p>

<div></div>

</body>

</html>

## **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

The following example shows some of the different speed curves that can be used:

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 50px;

background-color: red;

font-weight: bold;

position: relative;

animation: mymove 5s infinite;

}

#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;}

@keyframes mymove {

from {left: 0px;}

to {left: 300px;}

}

</style>

</head>

<body>

<h1>CSS Animation</h1>

<p>The animation-timing-function property specifies the speed curve of the animation. The following example shows some of the different speed curves that can be used:</p>

<div id="div1">linear</div>

<div id="div2">ease</div>

<div id="div3">ease-in</div>

<div id="div4">ease-out</div>

<div id="div5">ease-in-out</div>

</body>

</html>

## **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

The following example lets the <div> element retain the style values from the last keyframe when the animation ends:

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

position: relative;

animation-name: example;

animation-duration: 3s;

animation-fill-mode: forwards;

}

@keyframes example {

from {top: 0px;}

to {top: 200px; background-color: blue;}

}

</style>

</head>

<body>

<h1>CSS Animation</h1>

<p>Let the div element retain the style values set by the last keyframe when the animation ends:</p>

<div></div>

</body>

</html>

MOvebale letter Animation

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <title></title>

<style>

body

{

    background-color: gray;

}

ul{

    display: flex;

    list-style: none;

    margin-left: 550px;

    margin-top: 300px;

}

ul > li

{

    font-size: 60px;

    letter-spacing: 10px;

    font-family: sans-serif;

    color: white;

    text-shadow: 1px 1px 4px white;

    animation-name: apply;

    animation-duration: 4s;

    animation-iteration-count: infinite;

    font-weight: bold;

    animation-timing-function: ease-in;

}

@keyframes apply {

    0%{

        color: rgb(89, 0, 255);

    }

    25%

    {

        transform: translateX(-50px);

        letter-spacing: 35px;

        color: rgb(0, 255, 34);

        opacity: 0.5;

    }

    70%{

        transform: translateX(50px);

        color: rgb(247, 14, 169);

        opacity: 0.3;

    }

    100%

    {

        color: blue;

    }

}

ul > li:nth-child(1)

{

    animation-delay: 0s;

}

ul > li:nth-child(2)

{

    animation-delay: 0.4s;

}

ul > li:nth-child(3)

{

    animation-delay: 0.8s;

}

ul > li:nth-child(4)

{

    animation-delay: 1.2s;

}

ul > li:nth-child(5)

{

    animation-delay: 1.6s;

}

ul > li:nth-child(6)

{

    animation-delay: 2.0s;

}

ul > li:nth-child(7)

{

    animation-delay: 2.4s;

}

</style>

</head>

</head>

</head>

<body>

<ul>

<li>W</li>

<li>E</li>

<li>L</li>

<li>C</li>

<li>O</li>

<li>M</li>

<li>E</li>

</ul>

</body>

</html>

Hurt Animation

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <title>

    </title>

    <style type="text/css">

.heart

{

    border: 1px solid red;

    width: 150px;

    height: 150px;

    background-color: red;

    margin-left: 450px;

    margin-top: 300px;

    position: relative;

    transform: rotate(45deg);

    animation-name: heartbeat;

    animation-duration: 1.3s;

    animation-iteration-count: infinite;

}

 .heart::before

{

    position: absolute;

    content: "";

    border: 1px solid red;

    width: 150px;

    height: 150px;

    background-color: red;

    transform: translateY(-75px);

    border-radius: 50%;

}

.heart::after

{

    position: absolute;

    content: "";

    border: 1px solid red;

    width: 150px;

    height: 150px;

    background-color: red;

    transform: translateX(-75px);

    border-radius: 50%;

}

 @keyframes heartbeat {

    0%{transform: rotate(45deg) scale(1.4,1.4);}

    25%{transform: rotate(45deg) scale(1.2,1.2);}

    50%{transform: rotate(45deg) scale(1.4,1.4);}

    75%{transform: rotate(45deg) scale(1.2,1.2);}

    100%{transform: rotate(45deg) scale(1.4,1.4);}

}

    </style>

</head>

<body>

<div class="heart"></div>

</body>

</html>