# CSS3 Animations

CSS3 animations allows animation of most HTML elements without using JavaScript or Flash!

## What are CSS3 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 CSS3 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  element. The animation will last for 4 seconds, and it will gradually change the background-color of the  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;
}
```

**Note:** If the `animation-duration` property is not specified, the animation will have no effect, because the default value is 0.

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the  element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
/* The animation code */
@keyframes example {
    0%   {background-color: red;}
    25%  {background-color: yellow;}
    50%  {background-color: blue;}
    100% {background-color: green;}
}

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

The following example will change both the background-color and the position of the  element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
/* The animation code */
@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;}
}
/* The element to apply the animation to */
div {
    width: 100px;
    height: 100px;
    position: relative;
    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.

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;
}
```

# 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:

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: 3;
}
```

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

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

# Run Animation in Reverse Direction or Alternate Cycles

The `animation-direction` property is used to let an animation run in reverse direction or alternate cycles.

The following example will run the animation in reverse direction:

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: 3;
    animation-direction: reverse;
}
```

The following example uses the value "alternate" to make the animation first run forward, then backward, then forward:

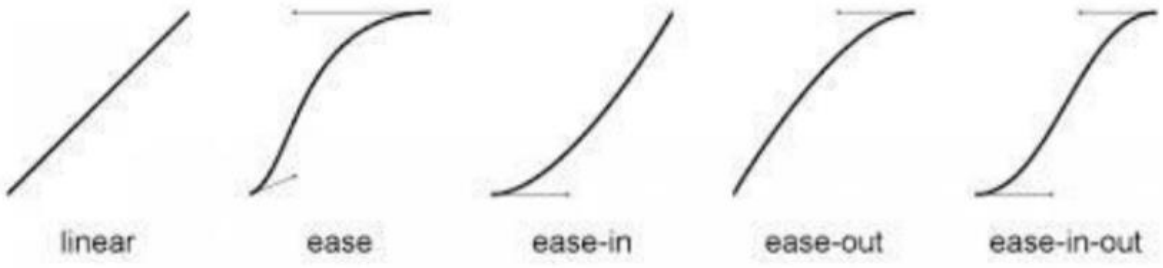
```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: 3;
    animation-direction: alternate;
}
```

# 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 the some of the different speed curves that can be used:

```
div {
    width: 100px;  height: 50px;
    background-color: red;    font-weight: bold;
    position: relative;
    animation: mymove 5s infinite;
}
/* Standard syntax */
#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;}

/* Standard syntax */
@keyframes mymove {
    from {left: 0px;}
    to {left: 300px;}
}
```

## Animation Shorthand Property

The example below uses six of the animation properties:

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

The same animation effect as above can be achieved by using the shorthand `animation` property:

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

## animation-play-state Property

The animation-play-state property specifies whether the animation is running or paused.
***Note:*** *Use this property in a JavaScript to pause an animation in the middle of a cycle.*

```
animation-play-state: paused|running|initial|inherit;
```

| Value | Description |
|---|---|
| Paused | Specifies that the animation is paused |
| Running | Default value. Specifies that the animation is running |

# animation-fill-mode Property

The animation-fill-mode property specifies a style for the element when the animation is not playing (when it is finished, or when it has a delay).

By default, CSS animations do not affect the element until the first keyframe is "played", and then stops once the last keyframe has completed. The animation-fill-mode property can override this behavior.

```
animation-fill-mode: none|forwards|backwards|both|initial|inherit;
```

| Value | Description |
|---|---|
| None | Default value. The animation will not apply any styles to the target element before or after it is executing |
| forwards | After the animation ends (determined by animation-iteration-count), the animation will apply the property values for the time the animation ended |
| backwards | The animation will apply the property values defined in the keyframe that will start the first iteration of the animation, during the period defined by animation-delay. These are either the values of the from keyframe (when animation-direction is "normal" or "alternate") or those of the to keyframe (when animation-direction is "reverse" or "alternate-reverse") |
| Both | The animation will follow the rules for both forwards and backwards. That is, it will extend the animation properties in both directions |

# Concatenate Animations

We want to animate a object, with 4 different set of keyframes, one after the other:

```
div#animated {
    animation-name: animation1, animation2, animation3;
    animation-duration: 12s, 4s, 3s;
    animation-delay: 0s, 12s, 16s; /* Animation waits the end of the preceding*/
    animation-fill-mode:  forwards, forwards, forward; /* fillmode forward */
    animation-timing-function: linear, ease, ease-in;
}
```