**Transition-property**

The transition-property CSS property sets the CSS properties to which a transition effect should be applied.

If you specify a shorthand property (e.g., background), all of its longhand sub-properties that can be animated will be.

Syntax

CSS

/\* Keyword values \*/

transition-property: none;

transition-property: all;

/\* <custom-ident> values \*/

transition-property: test\_05;

transition-property: -specific;

transition-property: sliding-vertically;

/\* Multiple values \*/

transition-property: test1, animation4;

transition-property: all, height, color;

transition-property: all, -moz-specific, sliding;

**Values**

none

No properties will transition.

all

All properties that can transition will.

<custom-ident>

A string identifying the property to which a transition effect should be applied when its value changes.

#### HTML

HTMLPlayCopy to Clipboard

<a class="target">Hover over me</a>

#### CSS

CSSPlayCopy to Clipboard

.target {

font-size: 14px;

transition-property: font-size;

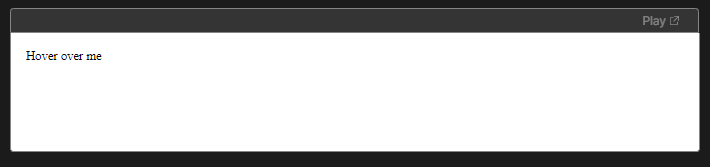
transition-duration: 4s;

}

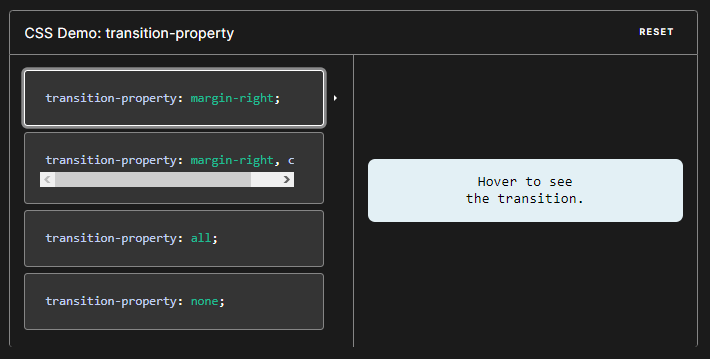
.target:hover {

font-size: 36px;

}



On hover the text-size will increase upto the given size



transition-property: margin-right;



transition-property: margin-right, color;

# Transition-duration

The transition-duration CSS property sets the length of time a transition animation should take to complete. By default, the value is 0s, meaning that no animation will occur.

You may specify multiple durations; each duration will be applied to the corresponding property as specified by the transition-property property, which acts as a master list. If the number of specified durations is less than in the master list, the user agent repeats the list of durations. If the number of specified durations is more than in the master list, the list is truncated to the right size. In both the cases, the CSS declaration stays valid.

**Syntax**

CSS

/\* <time> values \*/

transition-duration: 6s;

transition-duration: 120ms;

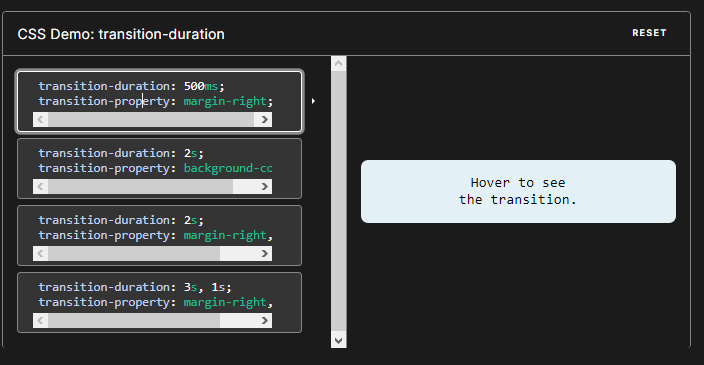
transition-duration: 1s, 15s;

transition-duration: 10s, 30s, 230ms;

**Values**

<time>

Is a <time> denoting the amount of time the transition from the old value of a property to the new value should take. A time of 0s indicates that no transition will happen, that is the switch between the two states will be instantaneous. A negative value for the time renders the declaration invalid.



1. transition-duration: 500ms;transition-property: margin-right;
2. transition-duration: 2s;transition-property: background-color;
3. transition-duration: 2s;transition-property: margin-right, color;
4. transition-duration: 3s, 1s;transition-property: margin-right, color;

Example showing different durations

HTML

<div class="box duration-1">0.5 seconds</div>

<div class="box duration-2">2 seconds</div>

<div class="box duration-3">4 seconds</div>

<button id="change">Change</button>

CSS

.box {

margin: 20px;

padding: 10px;

display: inline-block;

width: 100px;

height: 100px;

background-color: red;

font-size: 18px;

transition-property: background-color font-size transform color;

transition-timing-function: ease-in-out;

}

.transformed-state {

transform: rotate(270deg);

background-color: blue;

color: yellow;

font-size: 12px;

transition-property: background-color font-size transform color;

transition-timing-function: ease-in-out;

}

.duration-1 {

transition-duration: 0.5s;

}

.duration-2 {

transition-duration: 2s;

}

.duration-3 {

transition-duration: 4s;

}

JavaScript

function change() {

const elements = document.querySelectorAll("div.box");

for (const element of elements) {

element.classList.toggle("transformed-state");

}

}

const changeButton = document.querySelector("#change");

changeButton.addEventListener("click", change);







It will translate based on the given time

# Transition-timing-function

The transition-timing-function CSS property sets how intermediate values are calculated for CSS properties being affected by a transition effect.

This, in essence, lets you establish an acceleration curve so that the speed of the transition can vary over its duration.

This acceleration curve is defined using one <easing-function> for each property to be transitioned.

You may specify multiple easing functions; each one will be applied to the corresponding property as specified by the transition-property property, which acts as a transition-property list. If there are fewer easing functions specified than in the transition-property list, the user agent must calculate which value is used by repeating the list of values until there is one for each transition property. If there are more easing functions, the list is truncated to the right size. In both cases, the CSS declaration stays valid.

Syntax

CSS

/\* Keyword values \*/

transition-timing-function: ease;

transition-timing-function: ease-in;

transition-timing-function: ease-out;

transition-timing-function: ease-in-out;

transition-timing-function: linear;

transition-timing-function: step-start;

transition-timing-function: step-end;

/\* Function values \*/

transition-timing-function: steps(4, jump-end);

transition-timing-function: cubic-bezier(0.1, 0.7, 1, 0.1);

/\* Steps Function keywords \*/

transition-timing-function: steps(4, jump-start);

transition-timing-function: steps(10, jump-end);

transition-timing-function: steps(20, jump-none);

transition-timing-function: steps(5, jump-both);

transition-timing-function: steps(6, start);

transition-timing-function: steps(8, end);

/\* Multiple timing functions \*/

transition-timing-function: ease, step-start, cubic-bezier(0.1, 0.7, 1, 0.1);

**Values**

<easing-function>

Each <easing-function> represents the easing function to link to the corresponding property to transition, as defined in transition-property.

The non-step keyword values (ease, linear, ease-in-out, etc.) each represent cubic Bézier curve with fixed four point values, with the cubic-bezier() function value allowing for a non-predefined value. The step timing functions divides the input time into a specified number of intervals that are equal in length. It is defined by a number of steps and a step position.

ease

Equal to cubic-bezier(0.25, 0.1, 0.25, 1.0), the default value, increases in velocity towards the middle of the transition, slowing back down at the end.

linear

Equal to cubic-bezier(0.0, 0.0, 1.0, 1.0), transitions at an even speed.

ease-in

Equal to cubic-bezier(0.42, 0, 1.0, 1.0), starts off slowly, with the transition speed increasing until complete.

ease-out

Equal to cubic-bezier(0, 0, 0.58, 1.0), starts transitioning quickly, slowing down as the transition continues.

ease-in-out

Equal to cubic-bezier(0.42, 0, 0.58, 1.0), starts transitioning slowly, speeds up, and then slows down again.

cubic-bezier(p1, p2, p3, p4)

An author-defined cubic-Bezier curve, where the p1 and p3 values must be in the range of 0 to 1.

steps( n, <jumpterm>)

Displays the transition along n stops along the transition, displaying each stop for equal lengths of time. For example, if n is 5, there are 5 steps. Whether the transition holds temporarily at 0%, 20%, 40%, 60% and 80%, on the 20%, 40%, 60%, 80% and 100%, or makes 5 stops between the 0% and 100% along the transition, or makes 5 stops including the 0% and 100% marks (on the 0%, 25%, 50%, 75%, and 100%) depends on which of the following jump terms is used:

jump-start

Denotes a left-continuous function, so that the first jump happens when the transition begins;

jump-end

Denotes a right-continuous function, so that the last jump happens when the animation ends;

jump-none

There is no jump on either end. Instead, holding at both the 0% mark and the 100% mark, each for 1/n of the duration

jump-both

Includes pauses at both the 0% and 100% marks, effectively adding a step during the transition time.

start

Same as jump-start.

end

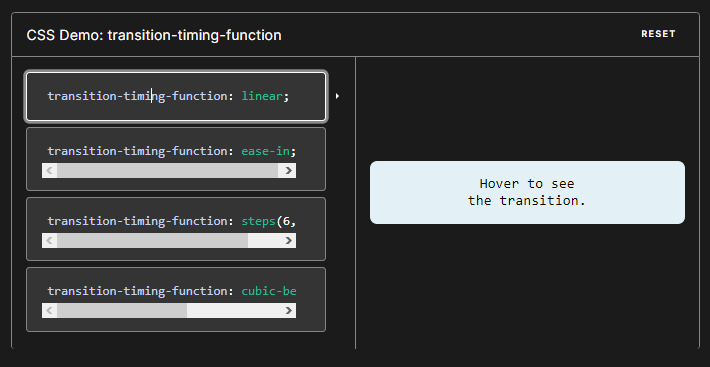
Same as jump-end.

step-start

Equal to steps(1, jump-start)

step-end

Equal to steps(1, jump-end)



1. transition-timing-function: linear;
2. transition-timing-function: ease-in;
3. transition-timing-function: steps(6, end);
4. transition-timing-function: cubic-bezier(.29, 1.01, 1, -0.68);

Examples

Cubic-Bezier examples

CSS

.ease {

transition-timing-function: ease;

}

.easein {

transition-timing-function: ease-in;

}

.easeout {

transition-timing-function: ease-out;

}

.easeinout {

transition-timing-function: ease-in-out;

}

.linear {

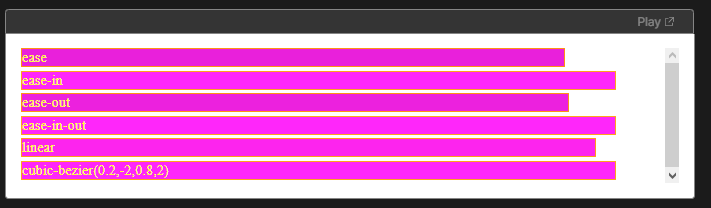
transition-timing-function: linear;

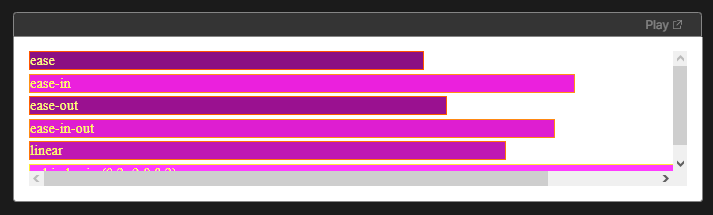
}

.cb {

transition-timing-function: cubic-bezier(0.2, -2, 0.8, 2);

}





# Transition-delay

The transition-delay CSS property specifies the duration to wait before starting a property's transition effect when its value changes.

The delay may be zero, positive, or negative:

A value of 0s (or 0ms) will begin the transition effect immediately.

A positive value will delay the start of the transition effect for the given length of time.

A negative value will begin the transition effect immediately, and partway through the effect. In other words, the effect will be animated as if it had already been running for the given length of time.

You may specify multiple delays, which is useful when transitioning multiple properties. Each delay will be applied to the corresponding property as specified by the transition-property property, which acts as a master list. If there are fewer delays specified than in the master list, the list of delay values will be repeated until there are enough. If there are more delays, the list of delay values will be truncated to match the number of properties. In both cases, the CSS declaration remains valid.

Syntax

CSS

/\* <time> values \*/

transition-delay: 3s;

transition-delay: 2s, 4ms;

Values

<time>

Denotes the amount of time to wait between a property's value changing and the start of the transition effect.





