**Transitions**

CSS transitions provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time. For example, if you change the color of an element from white to black, usually the change is instantaneous. With CSS transitions enabled, changes occur at time intervals that follow an acceleration curve, all of which can be customized.

Animations that involve transitioning between two states are often called implicit transitions as the states in between the start and final states are implicitly defined by the browser.

**Defining transitions**

CSS Transitions are controlled using the shorthand transition property. This is the best way to configure transitions, as it makes it easier to avoid out of sync parameters, which can be very frustrating to have to spend lots of time debugging in CSS.

You can control the individual components of the transition with the following sub-properties:

transition-property

Specifies the name or names of the CSS properties to which transitions should be applied. Only properties listed here are animated during transitions; changes to all other properties occur instantaneously as usual.

transition-duration

Specifies the duration over which transitions should occur. You can specify a single duration that applies to all properties during the transition, or multiple values to allow each property to transition over a different period of time.

transition-timing-function

Specifies a function to define how intermediate values for properties are computed. Timing functions determine how intermediate values of the transition are calculated. Most timing functions can be specified by providing the graph of the corresponding function, as defined by four points defining a cubic bezier. You can also choose easing from Easing Functions Cheat Sheet.

transition-delay

Defines how long to wait between the time a property is changed and the transition actually begins.

The transition shorthand CSS syntax is written as follows:

CSS

div {

transition: <property> <duration> <timing-function> <delay>;

}

Examples using js

JavaScript examples

### [Using transitions to make JavaScript functionality smooth](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_transitions/Using_CSS_transitions#using_transitions_to_make_javascript_functionality_smooth)

Transitions are a great tool to make things look much smoother without having to do anything to your JavaScript functionality. Take the following example.

HTMLCopy to Clipboard

<p>Click anywhere to move the ball</p>

<div id="foo" class="ball"></div>

Using JavaScript you can make the effect of moving the ball to a certain position happen:

JSCopy to Clipboard

const f = document.getElementById("foo");

document.addEventListener(

"click",

(ev) => {

f.style.transform = `translateY(${ev.clientY - 25}px)`;

f.style.transform += `translateX(${ev.clientX - 25}px)`;

},

false

);

With CSS you can make it smooth without any extra effort. Add a transition to the element and any change will happen smoothly:

CSSCopy to Clipboard

.ball {

border-radius: 25px;

width: 50px;

height: 50px;

background: #c00;

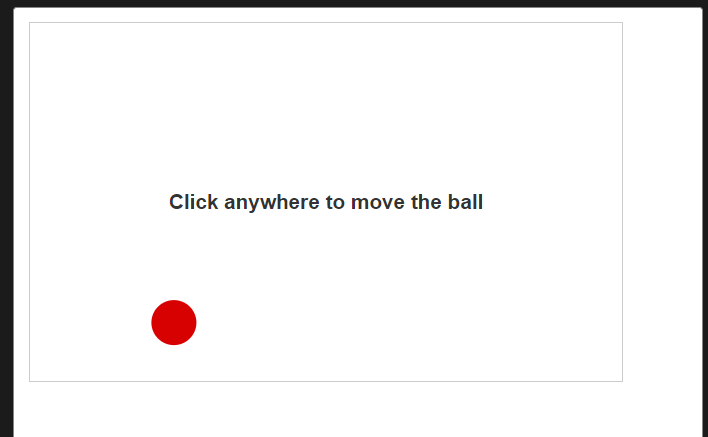
position: absolute;

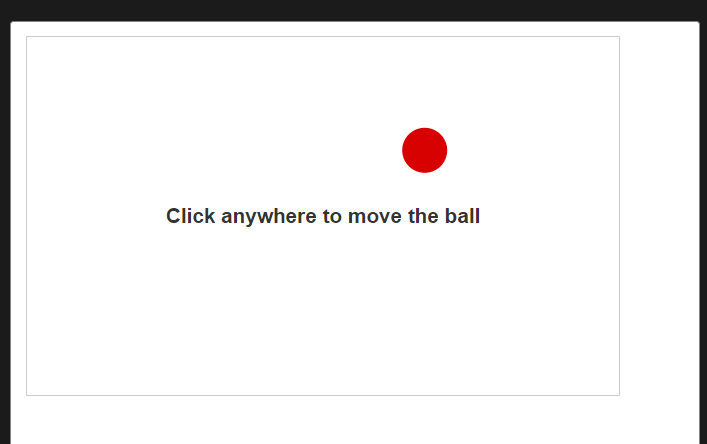
top: 0;

left: 0;

transition: transform 1s;

}





**Detecting the start and completion of a transition**

You can use the transitionend event to detect that an animation has finished running. This is a TransitionEvent object, which has two added properties beyond a typical Event object:

propertyName

A string indicating the name of the CSS property whose transition completed.

elapsedTime

A float indicating the number of seconds the transition had been running at the time the event fired. This value isn't affected by the value of transition-delay.

As usual, you can use the addEventListener() method to monitor for this event:

JS

el.addEventListener("transitionend", updateTransition, true);

You detect the beginning of a transition using transitionrun (fires before any delay) and transitionstart (fires after any delay), in the same kind of fashion:

JS

el.addEventListener("transitionrun", signalStart, true);

el.addEventListener("transitionstart", signalStart, true);