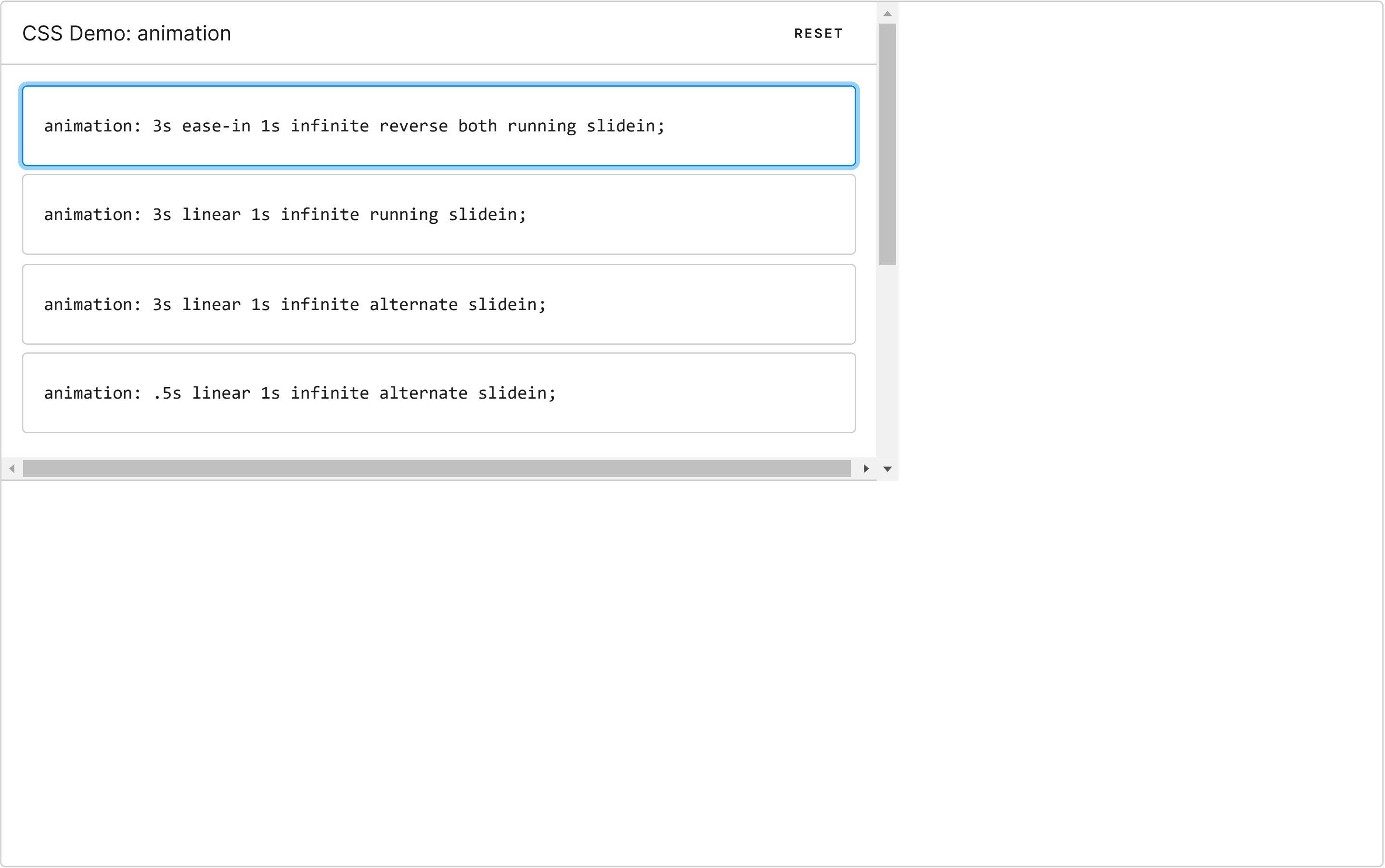
link: https://developer.mozilla.org/en-US/docs/Web/CSS/animation

animation

The `animation` shorthand CSS property applies an animation between styles. It is a shorthand for `animation-name`, `animation-duration`, `animation-timing-function`, `animation-delay`, `animation-iteration-count`, `animation-direction`, `animation-fill-mode`, and `animation-play-state`.

A description of which properties are animatable is available; it's worth noting that this description is also valid for CSS transitions.

Try it



CSS Demo: animation

```
animation: 3s ease-in 1s infinite reverse both running slidein;
```

```
animation: 3s linear 1s infinite running slidein;
```

```
animation: 3s linear 1s infinite alternate slidein;
```

```
animation: .5s linear 1s infinite alternate slidein;
```

Constituent properties

This property is a shorthand for the following CSS properties:

- `animation-delay`
- `animation-direction`
- `animation-duration`
- `animation-fill-mode`
- `animation-iteration-count`
- `animation-name`
- `animation-play-state`

- animation-timing-function

Syntax

```
/* @keyframes duration | easing-function | delay |
iteration-count | direction | fill-mode | play-state | name */
animation: 3s ease-in 1s 2 reverse both paused slidein;

/* @keyframes duration | easing-function | delay | name */
animation: 3s linear 1s slidein;

/* two animations */
animation: 3s linear slidein, 3s ease-out 5s slideout;
```

The `animation` property is specified as one or more single animations, separated by commas.

Each individual animation is specified as:

- zero, one, or two occurrences of the `<time>` value
- zero or one occurrences of the following values:
  - **`<single-easing-function>`**
  - **`<single-animation-iteration-count>`**
  - **`<single-animation-direction>`**
  - **`<single-animation-fill-mode>`**
  - **`<single-animation-play-state>`**
- an optional name for the animation, which may be `none` , a `<custom-ident>` , or a `<string>`

Values

`<single-easing-function>`

Determines the type of transition. The value must be one of those available in `<easing-function>` .

`<single-animation-iteration-count>`

The number of times the animation is played. The value must be one of those available in `animation-iteration-count` .

`<single-animation-direction>`

The direction in which the animation is played. The value must be one of those available in `animation-direction` .

`<single-animation-fill-mode>`

Determines how styles should be applied to the animation's target before and after its execution. The value must be one of those available in `animation-fill-mode` .

`<single-animation-play-state>`

Determines whether the animation is playing or not. The value must be one of those available in `animation-play-state` .

Description

The order of time values within each animation definition is important: the first value that can be parsed as a `<time>` is assigned to the `animation-duration` , and the second one is assigned to `animation-delay` .

The order of other values within each animation definition is also important for distinguishing an `animation-name` value from other values. If a value in the `animation` shorthand can be parsed as a value for an animation property other than `animation-name` , then the value will be applied to that property first and not to `animation-name` . For this reason, the recommended practice is to specify a value for `animation-name` as the last value in a list of values when using the `animation` shorthand; this holds true even when you specify multiple, comma-separated animations using the `animation` shorthand.

An `animation-name` value is not required to be declared in the `animation` shorthand property. If no name exists, there is no animation to apply on any of the properties.

When the `animation-duration` value is omitted from the `animation` shorthand property, the value for this property defaults to `0s` . In this case, the animation will still occur (the `animationStart` and `animationEnd` events will be fired) but no animation will be visible.

Accessibility concerns

Blinking and flashing animation can be problematic for people with cognitive concerns such as Attention Deficit Hyperactivity Disorder (ADHD). Additionally, certain kinds of motion can be a trigger for Vestibular disorders, epilepsy, and migraine and Scotopic sensitivity.

Consider providing a mechanism for pausing or disabling animation as well as using the Reduced Motion Media Query to create a complimentary experience for users who have expressed a preference for reduced animated experiences.

- Designing Safer Web Animation For Motion Sensitivity · An A List Apart Article
- An Introduction to the Reduced Motion Media Query | CSS-Tricks
- Responsive Design for Motion | WebKit
- MDN Understanding WCAG, Guideline 2.2 explanations
- Understanding Success Criterion 2.2.2 | W3C Understanding WCAG 2.0

Formal definition

Initial value	as each of the properties of the shorthand: • `animation-name` : `none` • `animation-duration` : `0s` • `animation-timing-function` : `ease` • `animation-delay` : `0s` • `animation-iteration-count` : `1` • `animation-direction` : `normal` • `animation-fill-mode` : `none` • `animation-play-state` : `running` • `animation-timeline` : `auto`
Applies to	all elements, `::before` and `::after` pseudo-elements
Inherited	no
Computed value	as each of the properties of the shorthand: • `animation-name` : as specified • `animation-duration` : as specified • `animation-timing-function` : as specified

- `animation-delay` : as specified
- `animation-direction` : as specified
- `animation-iteration-count` : as specified
- `animation-fill-mode` : as specified
- `animation-play-state` : as specified
- `animation-timeline` : listEachItemIdentifyerOrNoneAuto

Animation type	Not animatable

Formal syntax

```
animation = 
  <single-animation>#

<single-animation> = 
  <time>                              ||
  <easing-function>                   ||
  <time>                              ||
  <single-animation-iteration-count>  ||
  <single-animation-direction>        ||
  <single-animation-fill-mode>        ||
  <single-animation-play-state>       ||
  [ none | <keyframes-name> ]

<easing-function> = 
  linear                          |
  <linear-easing-function>        |
  <cubic-bezier-easing-function>  |
  <step-easing-function>

<single-animation-iteration-count> = 
  infinite        |
  <number [0,∞]>

<single-animation-direction> = 
  normal             |
  reverse            |
  alternate          |
  alternate-reverse

<single-animation-fill-mode> = 
  none       |
  forwards   |
  backwards  |
  both

<single-animation-play-state> = 
  running  |
  paused

<keyframes-name> = 
  <custom-ident>  |
  <string>

<linear-easing-function> = 
  linear( <linear-stop-list> )
```

```
<cubic-bezier-easing-function> = 
  ease                                                 |
  ease-in                                              |
  ease-out                                             |
  ease-in-out                                          |
  cubic-bezier( <number [0,1]> , <number> , <number [0,1]> , <number> )

<step-easing-function> = 
  step-start                                 |
  step-end                                   |
  steps( <integer> [, <step-position> ]? )

<linear-stop-list> = 
  [ <linear-stop> ]#

<step-position> = 
  jump-start  |
  jump-end    |
  jump-none   |
  jump-both   |
  start       |
  end

<linear-stop> = 
  <number>               &&
  <linear-stop-length>?

<linear-stop-length> = 
  <percentage>{1,2}
```

Examples

> **Note:** Animating CSS Box Model properties is discouraged. Animating any box model property is inherently CPU intensive; consider animating the transform property instead.

Sun Rise

Here we animate a yellow sun across a light blue sky. The sun rises to the center of the viewport and then falls out of sight.

```
<div class="sun"></div>
```

```
:root {
  overflow: hidden; /* hides any part of the sun below the horizon */
  background-color: lightblue;
  display: flex;
  justify-content: center; /* centers the sun in the background */
}

.sun {
  background-color: yellow;
  border-radius: 50%; /* creates a circular background */
  height: 100vh; /* makes the sun the size of the viewport */
  aspect-ratio: 1 / 1;
  animation: 4s linear 0s infinite alternate sun-rise;
}

@keyframes sun-rise {
  from {
    /* pushes the sun down past the viewport */
    transform: translateY(110vh);
  }
  to {
    /* returns the sun to its default position */
    transform: translateY(0);
  }
}
```

Animating Multiple Properties

Adding onto the sun animation in the previous example, we add a second animation changing the color of the sun as it rises and sets. The sun starts off dark red when it is below the horizon and changes to a bright orange as it reaches the top.

```
<div class="sun"></div>
```

```
:root {
  overflow: hidden;
  background-color: lightblue;
  display: flex;
  justify-content: center;
}

.sun {
  background-color: yellow;
  border-radius: 50%;
  height: 100vh;
  aspect-ratio: 1 / 1;
  animation: 4s linear 0s infinite alternate animating-multiple-properties;
}

/* it is possible to animate multiple properties in a single animation */
@keyframes animating-multiple-properties {
  from {
    transform: translateY(110vh);
    background-color: red;
    filter: brightness(75%);
  }
  to {
    transform: translateY(0);
    background-color: orange;
    /* unset properties i.e. 'filter' will revert to default values */
  }
}
```

Applying Multiple Animations

Here is a sun that rises and falls on a lightblue background. The sun gradually rotates through a rainbow of colors. The timing of the sun's position and color are independent.

```
<div class="sun"></div>
```

```
:root {
  overflow: hidden;
  background-color: lightblue;
  display: flex;
  justify-content: center;
}

.sun {
  background-color: yellow;
  border-radius: 50%;
  height: 100vh;
  aspect-ratio: 1 / 1;
  /* multiple animations are separated by commas */
  animation:
    4s linear 0s infinite alternate rise,
    /* animation parameters are set independently */
    24s linear 0s infinite psychedelic;
}

@keyframes rise {
  from {
    transform: translateY(110vh);
  }
  to {
    transform: translateY(0);
  }
}

@keyframes psychedelic {
  from {
    filter: hue-rotate(0deg);
  }
  to {
    filter: hue-rotate(360deg);
  }
}
```

Cascading Multiple Animations

Here is a yellow sun on a lightblue background. The sun bounces between the left and right sides of the viewport. The sun remains in the viewport even though a rise animation is defined. The rise animation's transform property is 'overwritten' by the bounce animation.

```html
<div class="sun"></div>
```

```css
:root {
  overflow: hidden;
  background-color: lightblue;
  display: flex;
  justify-content: center;
}

.sun {
  background-color: yellow;
  border-radius: 50%;
  height: 100vh;
  aspect-ratio: 1 / 1;
  /*
    animations declared later in the cascade will override the
    properties of previously declared animations
  */
  animation:
    4s linear 0s infinite alternate rise,
    /* bounce 'overwrites' the transform set by rise */
    4s linear 0s infinite alternate bounce;
    /* hence the sun only moves horizontally */
}

@keyframes rise {
  from {
    transform: translateY(110vh);
  }
  to {
    transform: translateY(0);
  }
}

@keyframes bounce {
  from {
    transform: translateX(-50vw);
  }
  to {
    transform: translateX(50vw);
  }
}
```

See Using CSS animations for additional examples.

Specifications

Specification
CSS Animations Level 1 # animation

Browser compatibility

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android
`animation`	Chrome 43	Edge 12	Firefox 16	Opera 30	Safari 9	Chrome 43 Android	Firefox 16 for Android

Tip: you can click/tap on a cell for more information.

mdn web docs_

See also

- Using CSS animations
- JavaScript `AnimationEvent` API

This page was last modified on Sep 28, 2022 by MDN contributors.