**Constituent properties**

## **CSS Syntax**

animation: *name duration timing-function delay iteration-count direction fill-mode play-state*;

**Animation-name**

The animation-name CSS property specifies the names of one or more @keyframes at-rules that describe the animation to apply to an element. Multiple @keyframe at-rules are specified as a comma-separated list of names. If the specified name does not match any @keyframe at-rule, no properties are animated.

**Syntax**

CSS

/\* Single animation \*/

animation-name: none;

animation-name: test\_05;

animation-name: -specific;

animation-name: sliding-vertically;

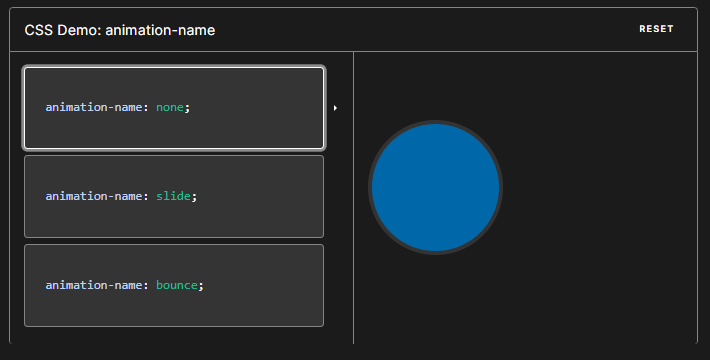
**Values**

1. none

A special keyword denoting no keyframes. It can be used to deactivate an animation without changing the ordering of the other identifiers, or to deactivate animations coming from the cascade.

1. <custom-ident>

A name identifying the animation. This identifier is composed of a combination of case-sensitive letters a to z, numbers 0 to 9, underscores (\_), and/or dashes (-). The first non-dash character must be a letter. Also, two dashes are forbidden at the beginning of the identifier. Furthermore, the identifier can't be none, unset, initial, or inherit.



**Examples**

Naming an animation

This animation has an animation-name of rotate.

CSS

.box {

background-color: rebeccapurple;

border-radius: 10px;

width: 100px;

height: 100px;

}

.box:hover {

animation-name: rotate;

animation-duration: 0.7s;

}

@keyframes rotate {

0% {

transform: rotate(0);

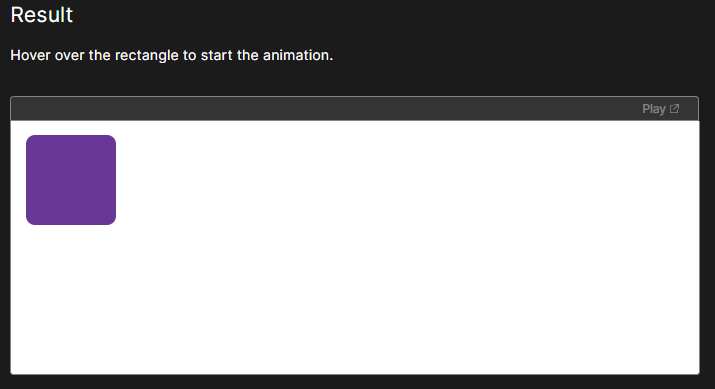
}

100% {

transform: rotate(360deg);

}

}



On hover it will rotate

# Animation-duration

The animation-duration CSS property sets the length of time that an animation takes to complete one cycle.

**Syntax**

/\* Single animation \*/

animation-duration: auto; /\* Default \*/

animation-duration: 6s;

animation-duration: 120ms;

Values

1. auto

For time-based animations, auto is equivalent to a value of 0s (see below). For CSS scroll-driven animations, auto fills the entire timeline with the animation.

1. <time>

The time that an animation takes to complete one cycle. This may be specified in either seconds (s) or milliseconds (ms). The value must be positive or zero and the unit is required.

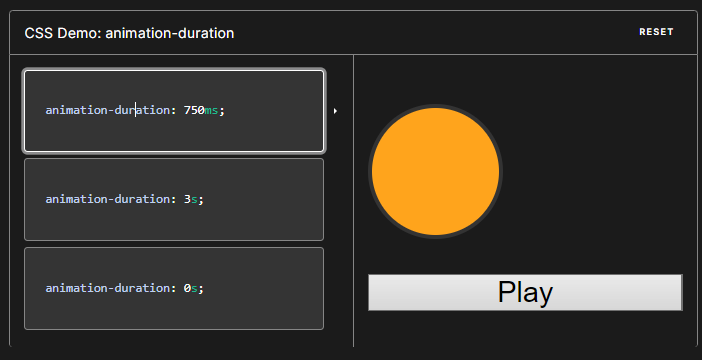
If no value is provided, the default value of 0s is used, in which case the animation still executes (the animationStart and animationEnd events are fired). Whether or not the animation will be visible when the duration is 0s will depend on the value of animation-fill-mode, as explained below:

If animation-fill-mode is set to backwards or both, the first frame of the animation as defined by animation-direction will be displayed during animation-delay countdown.

If animation-fill-mode is set to forwards or both, the last frame of the animation will be displayed, as defined by animation-direction, after the animation-delay expires.

If animation-fill-mode is set to none, the animation will have no visible effect.

Example:



CSS

.box {

background-color: rebeccapurple;

border-radius: 10px;

width: 100px;

height: 100px;

}

.box:hover {

animation-name: rotate;

animation-duration: 0.7s;

}

@keyframes rotate {

0% {

transform: rotate(0);

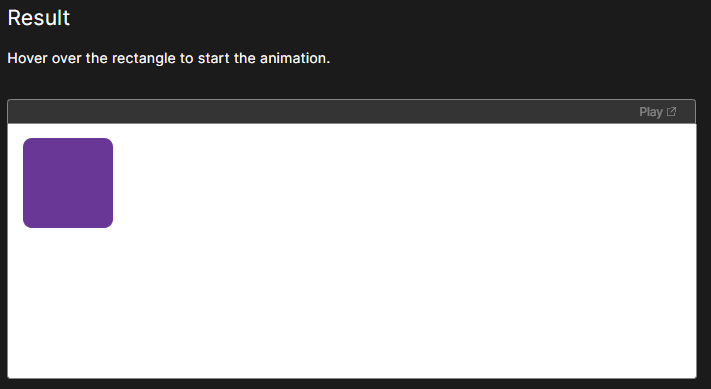
}

100% {

transform: rotate(360deg);

}

}



On hover it will rotate on the given duration

# Animation-timing-function

The animation-timing-function CSS property sets how an animation progresses through the duration of each cycle.

Syntax

CSS

/\* Keyword values \*/

animation-timing-function: ease;

animation-timing-function: ease-in;

animation-timing-function: ease-out;

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

animation-timing-function: linear;

animation-timing-function: step-start;

animation-timing-function: step-end;

**Values**

1. <easing-function>

The easing function that corresponds to a given animation, as determined by animation-name.

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.

1. ease

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

1. linear

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

1. ease-in

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

1. ease-out

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

1. ease-in-out

Equal to cubic-bezier(0.42, 0, 0.58, 1.0), with the animating properties slowly transitioning, speeding up, and then slowing down again.

1. 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.

1. steps(n, <jumpterm>)

Displays an animation iteration 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 animation 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 animation, 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:

1. jump-start

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

1. jump-end

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

1. 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.

1. jump-both

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

1. start

Same as jump-start.

1. end

Same as jump-end.

1. step-start

Equal to steps(1, jump-start)

1. step-end

Equal to steps(1, jump-end)

**Description**

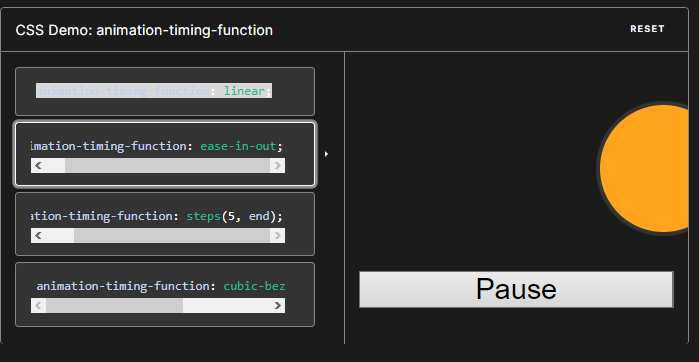
Timing functions may be specified on individual keyframes in a @keyframes rule. If no animation-timing-function is specified on a keyframe, the corresponding value of animation-timing-function from the element to which the animation is applied is used for that keyframe.

Within a keyframe, animation-timing-function is an at-rule-specific descriptor, not the property of the same name. The timing is not being animated. Rather, a keyframe's timing function is applied on a property-by-property basis from the keyframe on which it is specified until the next keyframe specifying that property, or until the end of the animation if there is no subsequent keyframe specifying that property. As a result, an animation-timing-function specified on the 100% or to keyframe will never be used.

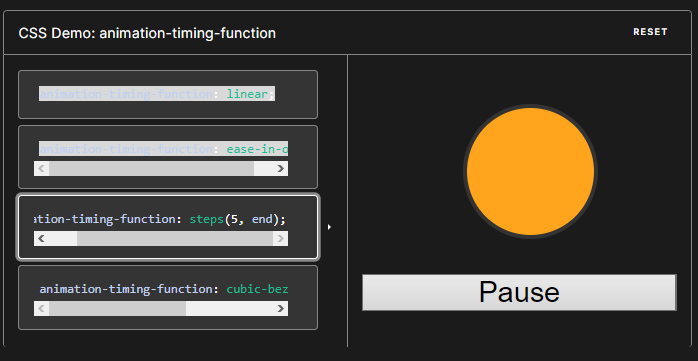
Example:



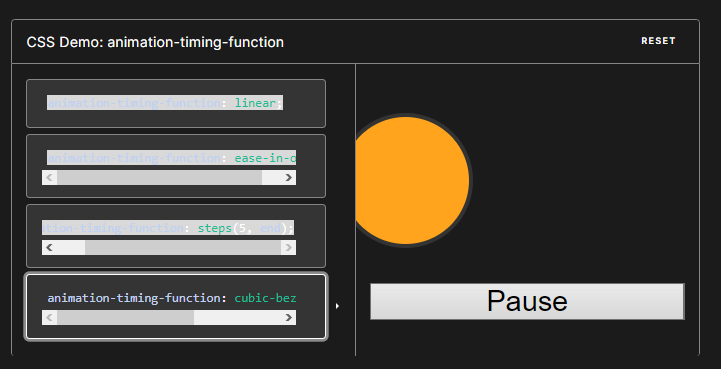
animation-timing-function: linear;



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



animation-timing-function: steps(5, end);



animation-timing-function: cubic-bezier(0.1, -0.6, 0.2, 0);

# Animation-delay

The animation-delay CSS property specifies the amount of time to wait from applying the animation to an element before beginning to perform the animation. The animation can start later, immediately from its beginning, or immediately and partway through the animation.

Syntax

CSS

/\* Single animation \*/

animation-delay: 3s;

animation-delay: 0s;

animation-delay: -1500ms;

/\* Multiple animations \*/

animation-delay: 2.1s, 480ms;

**Values**

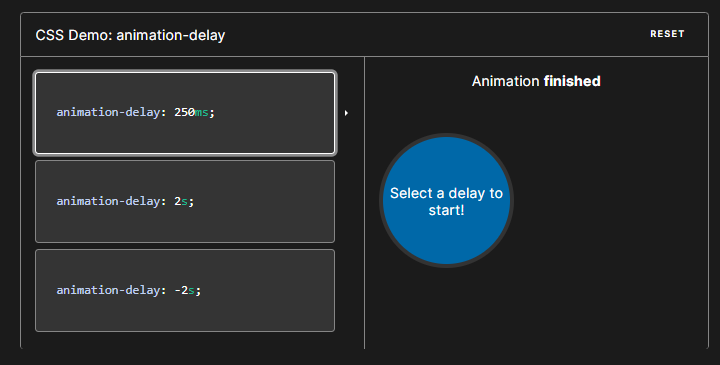
<time>

The time offset, from the moment at which the animation is applied to the element, at which the animation should begin. This may be specified in either seconds (s) or milliseconds (ms). The unit is required.

A positive value indicates that the animation should begin after the specified amount of time has elapsed. A value of 0s, which is the default, indicates that the animation should begin as soon as it's applied.

A negative value causes the animation to begin immediately, but partway through its cycle. For example, if you specify -1s as the animation delay time, the animation will begin immediately but will start 1 second into the animation sequence. If you specify a negative value for the animation delay, but the starting value is implicit, the starting value is taken from the moment the animation is applied to the element.

Example:



# Animation-iteration-count

The animation-iteration-count CSS property sets the number of times an animation sequence should be played before stopping.

Syntax

CSS

/\* Keyword value \*/

animation-iteration-count: infinite;

/\* <number> values \*/

animation-iteration-count: 3;

animation-iteration-count: 2.4;

**Values**

Infinite - The animation will repeat forever.

<number> - The number of times the animation will repeat; this is 1 by default. You may specify non-integer values to play part of an animation cycle: for example, 0.5 will play half of the animation cycle. Negative values are invalid.

Example:

#### CSS

CSSPlayCopy to Clipboard

.box {

background-color: rebeccapurple;

border-radius: 10px;

width: 100px;

height: 100px;

}

.box:hover {

animation-name: rotate;

animation-duration: 0.7s;

animation-iteration-count: 10;

}

@keyframes rotate {

0% {

transform: rotate(0);

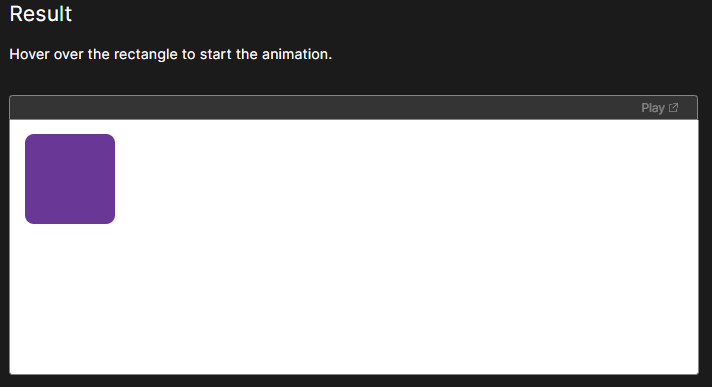
}

100% {

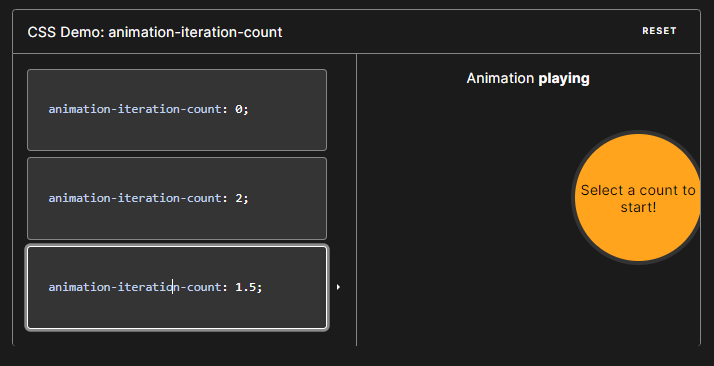
transform: rotate(360deg);

}

}



On hover the box will rotate 2x times



# Animation-direction

The animation-direction CSS property sets whether an animation should play forward, backward, or alternate back and forth between playing the sequence forward and backward.

Syntax

CSS

/\* Single animation \*/

animation-direction: normal;

animation-direction: reverse;

animation-direction: alternate;

animation-direction: alternate-reverse;

/\* Multiple animations \*/

animation-direction: normal, reverse;

animation-direction: alternate, reverse, normal;

**Values**

normal

The animation plays forwards each cycle. In other words, each time the animation cycles, the animation will reset to the beginning state and start over again. This is the default value.

reverse

The animation plays backwards each cycle. In other words, each time the animation cycles, the animation will reset to the end state and start over again. Animation steps are performed backwards, and timing functions are also reversed. For example, an ease-in timing function becomes ease-out.

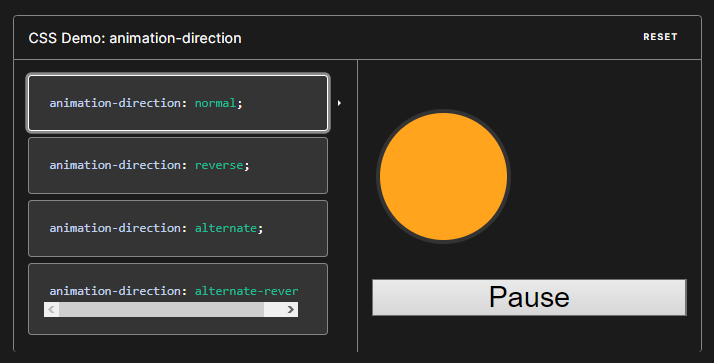
alternate

The animation reverses direction each cycle, with the first iteration being played forwards. The count to determine if a cycle is even or odd starts at one.

alternate-reverse

The animation reverses direction each cycle, with the first iteration being played backwards. The count to determine if a cycle is even or odd starts at one.

Example:



#### CSS

CSSPlayCopy to Clipboard

.box {

background-color: rebeccapurple;

border-radius: 10px;

width: 100px;

height: 100px;

}

.box:hover {

animation-name: rotate;

animation-duration: 0.7s;

animation-direction: reverse;

}

@keyframes rotate {

0% {

transform: rotate(0);

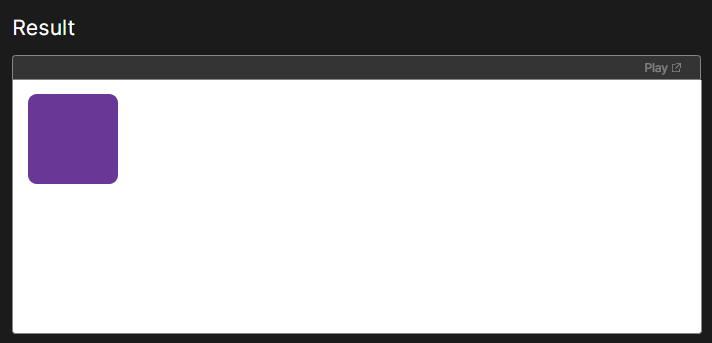
}

100% {

transform: rotate(360deg);

}

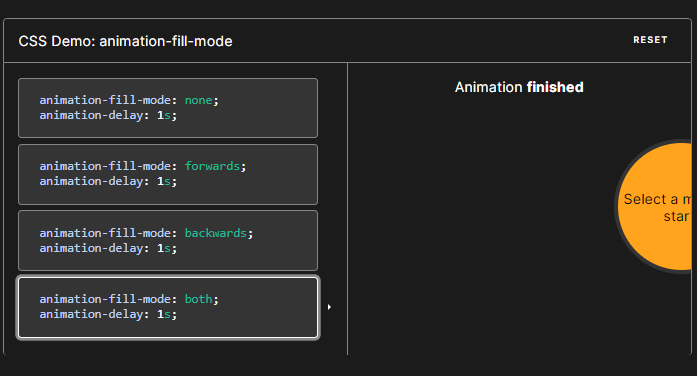
}



On hover the box will rotate reverse

# Animation-fill-mode

The animation-fill-mode CSS property sets how a CSS animation applies styles to its target before and after its execution.



Syntax

CSS

/\* Single animation \*/

animation-fill-mode: none;

animation-fill-mode: forwards;

animation-fill-mode: backwards;

animation-fill-mode: both;

/\* Multiple animations \*/

animation-fill-mode: none, backwards;

animation-fill-mode: both, forwards, none;

Values

none

The animation will not apply any styles to the target when it's not executing. The element will instead be displayed using any other CSS rules applied to it. This is the default value.

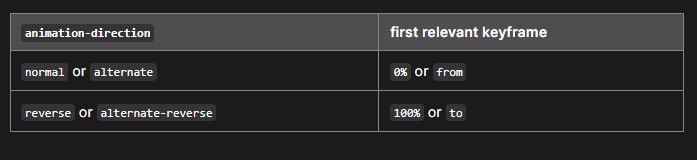
forwards

The target will retain the computed values set by the last keyframe encountered during execution. The last keyframe depends on the value of animation-direction and animation-iteration-count:



backwards

The animation will apply the values defined in the first relevant keyframe as soon as it is applied to the target, and retain this during the animation-delay period. The first relevant keyframe depends on the value of animation-direction:



both

The animation will follow the rules for both forwards and backwards, thus extending the animation properties in both directions.

HTMLPlayCopy to Clipboard

<p>Move your mouse over the gray box!</p>

<div class="demo">

<div class="growsandstays">This grows and stays big.</div>

<div class="grows">This just grows.</div>

</div>

#### CSS

CSSPlayCopy to Clipboard

.demo {

border-top: 100px solid #ccc;

height: 300px;

}

@keyframes grow {

0% {

font-size: 0;

}

100% {

font-size: 40px;

}

}

.demo:hover .grows {

animation-name: grow;

animation-duration: 3s;

}

.demo:hover .growsandstays {

animation-name: grow;

animation-duration: 3s;

animation-fill-mode: forwards;

}

# Animation-play-state

The animation-play-state CSS property sets whether an animation is running or paused.



Resuming a paused animation will start the animation from where it left off at the time it was paused, rather than starting over from the beginning of the animation sequence.

Syntax

CSS

/\* Single animation \*/

animation-play-state: running;

animation-play-state: paused;

/\* Multiple animations \*/

animation-play-state: paused, running, running;

Values

running

The animation is currently playing.

paused

The animation is currently paused.

HTMLPlayCopy to Clipboard

<div class="box"></div>

#### CSS

CSSPlayCopy to Clipboard

.box {

background-color: rebeccapurple;

border-radius: 10px;

width: 100px;

height: 100px;

animation-name: rotate;

animation-duration: 0.7s;

animation-iteration-count: infinite;

animation-play-state: paused;

}

.box:hover {

animation-play-state: running;

}

@keyframes rotate {

0% {

transform: rotate(0);

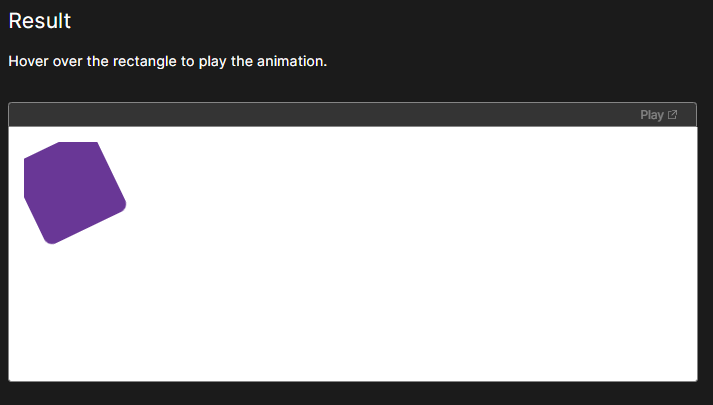
}

100% {

transform: rotate(360deg);

}

}



Only on hover the box will rotate otherwise it will pause

# Animation-timeline

The animation-timeline CSS property specifies the timeline that is used to control the progress of a CSS animation.

The following types of timelines can be set via animation-timeline:

The default document timeline, which is progressed through by the passing of time since the document was first loaded in the browser. This is the timeline traditionally associated with CSS animations and is selected with a value of auto, or by not specifying an animation-timeline value at all.

A scroll progress timeline, which is progressed through by scrolling a scrollable element (scroller) between top and bottom (or left and right). The position in the scroll range is converted into a percentage of progress — 0% at the start and 100% at the end. The element that provides the scroll progress timeline can be specified in two ways:

A named scroll progress timeline is one where the scroller providing the scroll progress timeline is explicitly named using the scroll-timeline-name property (or the scroll-timeline shorthand property). The name is then linked to the element to animate by specifying it as the value of that element's animation-timeline property.

An anonymous scroll progress timeline is one where the element to animate is given a scroll() function as an animation-timeline value, which selects the scroller providing the scroll progress timeline and the scroll axis to be used based on the arguments you pass to it.

A view progress timeline, which is progressed through based on the change in visibility of an element (known as the subject) inside a scroller. The visibility of the subject inside the scroller is tracked — by default, the timeline is at 0% when the subject is first visible at one edge of the scroller, and 100% when it reaches the opposite edge. Unlike with scroll progress timelines, you can't specify the scroller — the subject's visibility is always tracked within its nearest ancestor scroller. The subject that provides the view progress timeline can be specified in two ways:

A named view progress timeline is one where the subject is explicitly named using the view-timeline-name property (or the view-timeline shorthand property). The name is then linked to the element to animate by specifying it as the value of that element's animation-timeline property. This is a key point — with named view progress timelines, the element to animate does not have to be the same as the subject.

An anonymous view progress timeline is one where the subject is given a view() function as an animation-timeline value, causing it to be animated based on its position inside its nearest parent scroller.

Syntax

CSS

/\* Keyword \*/

animation-timeline: none;

animation-timeline: auto;

/\* Single animation named timeline \*/

animation-timeline: --timeline\_name;

/\* Single animation anonymous scroll progress timeline \*/

animation-timeline: scroll();

animation-timeline: scroll(scroller axis);

/\* Single animation anonymous view progress timeline \*/

animation-timeline: view();

animation-timeline: view(axis inset);

/\* Multiple animations \*/

animation-timeline: test1, animation4;

animation-timeline: none, -moz-specific, sliding;

Values

none

The animation is not associated with a timeline.

auto

The animation's timeline is the document's default DocumentTimeline.

scroll()

An anonymous scroll progress timeline is provided by some ancestor scroller of the current element. The function parameters allow you to select the scroller, and the scrolling axis the timeline will be measured along.

See scroll() for more information.

view()

An anonymous view progress timeline is provided by the subject that animation-timeline: view(); is set on. The function parameters allow you to select the scrollbar axis along which timeline progress will be tracked and an inset that adjusts the position of the box in which the subject is deemed to be visible.

See view() for more information.

<timeline-name>

A <custom-ident> or string identifying a named timeline previously declared with the scroll-timeline-name property (or scroll-timeline shorthand property). The name must begin with --.

Example:

### [Setting a named scroll progress timeline](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-timeline#setting_a_named_scroll_progress_timeline)

A scroll progress timeline named --squareTimeline is defined using the scroll-timeline-name property on an element with an id of container. This is then set as the timeline for the animation on the #square element using animation-timeline: --squareTimeline.

#### HTML

The HTML for the example is shown below.

HTMLPlayCopy to Clipboard

<div id="container">

<div id="square"></div>

<div id="stretcher"></div>

</div>

#### CSS

The CSS for the container sets it as the source of a scroll progress timeline named --squareTimeline using the scroll-timeline-name property (we could explicitly set which scrollbar axis to use with [scroll-timeline-axis](https://developer.mozilla.org/en-US/docs/Web/CSS/scroll-timeline-axis), but there is only a block direction scrollbar here, and it will be used by default).

The height of the container is set to 300px and we also set the container to create a vertical scrollbar if it overflows (below we will use CSS on the "stretcher" element to ensure that it does overflow).

CSSPlayCopy to Clipboard

#container {

height: 300px;

overflow-y: scroll;

scroll-timeline-name: --squareTimeline;

position: relative;

}

The CSS below defines a square that rotates in alternate directions according to the timeline provided by the animation-timeline property, which is set to the --squareTimeline timeline named above.

CSSPlayCopy to Clipboard

#square {

background-color: deeppink;

width: 100px;

height: 100px;

margin-top: 100px;

animation-name: rotateAnimation;

animation-duration: 1ms; /\* Firefox requires this to apply the animation \*/

animation-direction: alternate;

animation-timeline: --squareTimeline;

position: absolute;

bottom: 0;

}

@keyframes rotateAnimation {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

The "stretcher" CSS sets the block height to 600px, which forces the container element to overflow and create scroll bars. Without this element there would be no scrollbar, and hence no scroll progress timeline to associate with the animation timeline.

CSSPlayCopy to Clipboard

#stretcher {

height: 600px;

}

This property is in-development