

# CSS-анимация

---

# Ключевые кадры

---

Анимация позволяет элементу постепенно переходить от одного стиля к другому.

Чтобы использовать анимацию CSS, необходимо сначала указать некоторые ключевые кадры для анимации.

Ключевые кадры указывают какие стили будут у элемента в определенное время.

Ключевые кадры указываются с помощью правила `@keyframes`, определяемого следующим образом:

`@keyframes` имя анимации { список правил }

Создание анимации начинается с установки ключевых кадров правила `@keyframes`.

Кадры определяют, какие свойства на каком шаге будут анимированы.

При указании стилей CSS внутри `@keyframes` правило, анимация будет постепенно меняться от текущего стиля к новому стилю в определенное время.

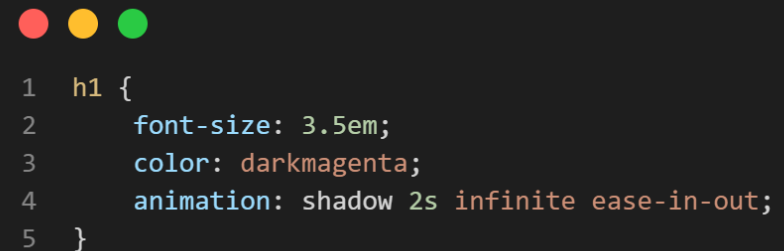
Ключевые кадры создаются с помощью ключевых слов `from` и `to` (эквивалентны значениям `0%` и `100%`) или с помощью процентных пунктов, которых можно задавать сколько угодно. Также можно комбинировать ключевые слова и процентные пункты.

Если `0%` или `100%` кадры не указаны, то браузер пользователя создает их, используя вычисляемые (первоначально заданные) значения анимируемого свойства.

```
1  @keyframes shadow {
2    from {
3      text-shadow: 0 0 3px black;
4    }
5
6    50% {
7      text-shadow: 0 0 30px black;
8    }
9
10   to {
11     text-shadow: 0 0 3px black;
12   }
13 }
```

После объявления правила `@keyframes`, мы можем ссылаться на него в свойстве `animation`.

`animation` - универсальное свойство, которое задаёт сразу несколько параметров анимации.



```
1 h1 {  
2   font-size: 3.5em;  
3   color: darkmagenta;  
4   animation: shadow 2s infinite ease-in-out;  
5 }
```

Все параметры воспроизведения анимации можно объединить в одном свойстве — `animation`, перечислив их через пробел:

`animation: animation-name animation-duration animation-timing-function animation-delay animation-iteration-count animation-direction;`

Для воспроизведения анимации достаточно указать только два свойства — `animation-name` и `animation-duration`, остальные свойства примут значения по умолчанию. Порядок перечисления свойств не имеет значения, единственное, время выполнения анимации `animation-duration` обязательно должно стоять перед задержкой `animation-delay`

# Значение по умолчанию

---

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-name

---

Устанавливает одну или несколько анимаций, которые применяются к элементу. Представляет собой имя, связанное с правилом @keyframes, оно в свою очередь задаёт последовательность движения.

**none** - отключает анимацию.

**<идентификатор>** - текстовая строка, которая связана с анимацией. Идентификатор должен начинаться с латинской буквы или подчёркивания (\_), также может содержать цифры от 0 до 9 и дефис (-). Запрещено использовать зарезервированные ключевые слова none, unset, inherit, initial.

## animation-duration

---

Задаёт время в секундах или миллисекундах, сколько должен длиться один цикл анимации. По умолчанию значение равно 0s, это означает, что никакой анимации нет.

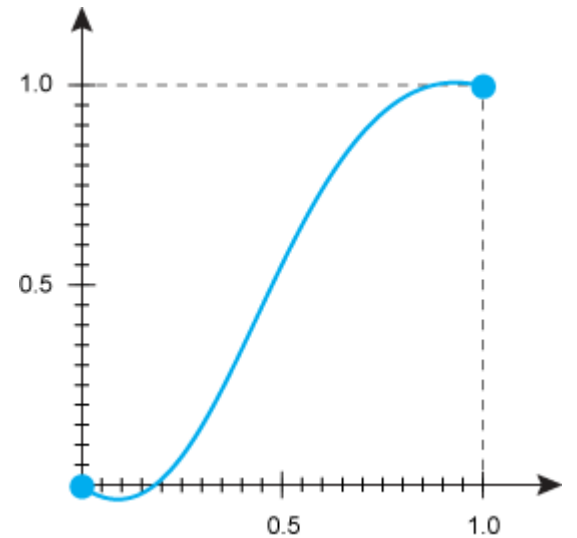
Можно указать несколько значений, перечисляя их через запятую. Отрицательные значения и значения без указания единиц времени (s или ms) недопустимы и будут игнорироваться.

`animation-duration: 1s;`

## animation-timing-function

Устанавливает, согласно какой функции времени должна происходить анимация каждого цикла между ключевыми кадрами. Она представляет собой математическую функцию, показывающую, как быстро по времени меняется значение свойства. Начальная точка имеет координаты 0.0, 0.0, конечная — 1.0, 1.0, при этом функция по оси ординат может превышать эти значения в большую или меньшую сторону (рис. 1).

**animation-timing-function:** ease | ease-in | ease-out | ease-in-out | linear | step-start | step-end | steps | cubic-bezier





# Значения animation-timing-function:

**ease** - анимация начинается медленно, затем ускоряется и к концу движения опять замедляется. Аналогично `cubic-bezier(0.25,0.1,0.25,1)`.

**ease-in** - анимация медленно начинается, к концу ускоряется. Аналогично `cubic-bezier(0.42,0,1,1)`.

**ease-out** - анимация начинается быстро, к концу замедляется. Аналогично `cubic-bezier(0,0,0.58,1)`.

**ease-in-out** - анимация начинается и заканчивается медленно. Аналогично `cubic-bezier(0.42,0,0.58,1)`.

**linear** - одинаковая скорость от начала и до конца.

**step-start** - как таковой анимации нет. Стиливые свойства сразу же принимают конечное значение.

**step-end** - как таковой анимации нет. Стиливые свойства находятся в начальном значении заданное время, затем сразу же принимают конечное значение.

**steps** - ступенчатая функция, имеющая заданное число шагов.

`animation-timing-function: steps(<число>, start | end)`

Здесь: <число> — целое число больше нуля;

Со значением `start` анимация начинается в начале каждого шага, со значением `end` — в конце каждого шага с задержкой. Задержка вычисляется как результат деления времени анимации на количество шагов. Если второй параметр не указан, используется значение по умолчанию `end`.

**cubic-bezier** - задаёт функцию движения в виде кривой Безье

## animation-delay

---

Свойство `animation-delay` устанавливает время ожидания перед запуском цикла анимации. Значение `0s` или `0ms` запускает анимацию сразу же. Отрицательное значение также включает анимацию без задержек, но может привести к изменению вида начала анимации.

Допустимо указывать несколько значений, перечисляя их через запятую.

`animation-delay: 1s;`

## animation-iteration-count

---

Свойство определяет, сколько раз проигрывать цикл анимации до её остановки.

Значения:

**infinite** - Анимация повторяется бесконечно.

**<число>** - Сколько раз должна повторяться анимация. Отрицательные значения не допустимы. Можно использовать числа меньше единицы, для примера 0.5 будет означать половину цикла анимации.

# animation-direction

---

Устанавливает направление движения анимации.

## Значения

**normal** - анимация идёт с самого начала, после завершения цикла возвращается к исходному состоянию.

**alternate** - анимация идёт с начала до конца, затем плавно возвращается в исходное положение.

**reverse** - анимация идёт с конца цикла, после его завершения возвращается к исходному состоянию.

**alternate-reverse** - анимация идёт с конца до начала, затем плавно возвращается в исходное положение.

# animation-fill-mode

---

Определяет, какие стили должны применяться к элементу, когда анимация не проигрывается. Например, после её завершения или при остановке. По умолчанию, в момент окончания анимации стиль элемента возвращается к исходному, свойство `animation-fill-mode` позволяет изменить это поведение и сделать так, чтобы стиль элемента оставался как у последнего ключевого кадра.

К примеру, если вы делаете выезжающее из-за края окна браузера сообщение, то после окончания анимации сообщение вернётся обратно за край экрана. Значение `forwards` свойства `animation-fill-mode` изменяет это поведение и оставляет стили на момент завершения движения. Таким образом, сообщение плавно выдвинется из-за края окна и останется на месте.

## Значения animation-fill-mode:

---

**none** - к элементу не применяются какие-либо стили.

**forwards** - к элементу по окончании анимации применяется стиль последнего ключевого кадра. Каким будет этот кадр в последовательности анимации зависит от комбинации значений свойств **animation-direction** и **animation-iteration-count**.

**backwards** - к элементу применяется стиль первого ключевого кадра и он остаётся на протяжении периода заданного **animation-delay**. Первый ключевой кадр определяется на основании значения **animation-direction**.

**both** - к элементу применяются оба правила, как для **forwards**, так и для **backwards**.

## animation-play-state

---

Свойство определяет, проигрывать анимацию или поставить её на паузу. При продолжении установленной на паузе анимации она начинается с того момента где была остановлена.

**running** - проигрывать анимацию.

**paused** - поставить анимацию на паузу.