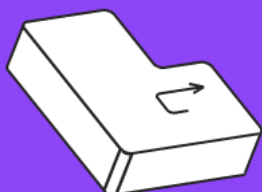




# Урок 7. Псевдоклассы и псевдоэлементы

Веб-вёрстка HTML / CSS



# Оглавление

Псевдоклассы и псевдоэлементы	3
Псевдоклассы	3
Псевдоклассы, определяющие состояние элементов	4
Структурные псевдоклассы	5
Псевдоклассы по типу дочернего элемента	5
Оформление результатов проверки	5
Пример использования псевдоклассов	6
Комбинирование псевдоклассов	7
Псевдоэлементы	7
Пример использования псевдоэлементов	8
Выводы работы псевдоклассов и псевдоэлементов.	9
Использование SVG в HTML5	
Использование <object> тега	10
Использование <iframe> тега	10
Использование <img> тега	11
Использование <svg> тега	11
Использование CSS Background Image	12
Как добавить эффект наведения на svg изображение	12
CSS3 Трансформация	13
CSS3 transition	15
CSS3 Анимация	16
Длительность анимации animation-duration	17
Временная функция animation-timing-function	18
Шаги анимации	19
Анимация с задержкой animation-delay	20
Повтор анимации animation-iteration-count	21
Направление анимации animation-direction	21
Проигрывание анимации animation-play-state	22

Состояние элемента до и после воспроизведения анимации animation-fill-mode	22
Краткая запись анимации	23
Множественные анимации	23

## Псевдоклассы и псевдоэлементы

Существуют стили и элементы, которые мы встречаем на каждой странице, но они появляются не сразу, а в момент взаимодействия с пользователем, хорошим примером будет эффект наведения, меню нашего сайта выглядит так как в макете, а при наведении на элементы меню они чаще всего меняют значение цвета текста или фона. Получается нам необходимо добавить не просто класс, а псевдокласс, тот стиль, который будет обрабатывать в определенный момент времени.

## Псевдоклассы

С псевдоклассами добавляются особые классы к элементам. Выбираются объекты, которые:

- отсутствуют в структуре веб-страницы;
- нельзя выбрать с помощью обычных селекторов. Например, первая буква или первая строка одного абзаца.

Вы наверняка замечали на сайтах: когда наводите мышкой на конкретный пункт меню, он меняет свой вид. У него изменяется цвет фона, цвет ссылки, даже шрифт или его размер. Это происходит благодаря псевдоклассам. Рассмотрим их синтаксис.

```
селектор:псевдокласс {  
    свойство: значение;  
}  
  
a:hover {  
    color: #ccc;  
}
```

После селектора ставится двоеточие. Сразу после него без пробела указывается название псевдокласса.

## Псевдоклассы, определяющие состояние элементов

Начнём с самых часто используемых псевдоклассов, определяющие состояние, простыми словами мы меняем состояние элемента к которому добавляем псевдокласс.

1. `a:link` — ссылается на посещенную ссылку.
2. `a:visited` — ссылается на уже посещенную ссылку.
3. `a:hover` — ссылается на любой элемент, по которому проводят курсором мыши.
4. `a:focus` — ссылается на любой элемент, над которым находится курсор мыши.
5. `a:active` — ссылается на активированный пользователем элемент.
6. `:valid` — выберет поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу.
7. `:invalid` — выберет поля формы, содержимое которых не соответствует указанному типу.
8. `:enabled` — выберет все доступные (активные) поля форм.
9. `:disabled` — выберет заблокированные поля форм, т. е. находящиеся в неактивном состоянии.
10. `:in-range` — выберет поля формы, значения которых находятся в заданном диапазоне.
11. `:out-of-range` — выберет поля формы, значения которых не входят в установленный диапазон.
12. `:lang()` — выбирает абзацы на указанном языке.
13. `:not(селектор)` — выберет элементы, которые не содержат указанный селектор. Например, класс, идентификатор или селектор элемента `:not([type="submit"])`.
14. `:checked` — выбирает выделенные (выбранные пользователем) элементы.

# Структурные псевдоклассы

1. `:nth-child(odd)` — выбирает нечётные дочерние элементы.
2. `:nth-child(even)` — выбирает чётные дочерние элементы.
3. `:nth-child(3n)` — выбирает каждый третий элемент среди дочерних.
4. `:nth-child(3n+2)` — выбирает каждый третий элемент, начиная со второго дочернего элемента (+2).
5. `:nth-child(n+2)` — выбирает все элементы, начиная со второго.
6. `:nth-child(3)` — выбирает третий дочерний элемент.
7. `:nth-last-child()` — в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с `:nth-child()`, но начиная с последнего, в обратную сторону.
8. `:first-child` — позволяет оформить только самый первый дочерний элемент тега.
9. `:last-child` — позволяет форматировать последний дочерний элемент тега.
10. `:only-child` — выбирает элемент, являющийся единственным дочерним элементом.
11. `:empty` — выбирает элементы, у которых нет дочерних элементов.
12. `:root` — выбирает корневой элемент в документе (элемент `html`).

## Псевдоклассы по типу дочернего элемента

1. `:nth-of-type()` — выбирает элементы по аналогии с `:nth-child()`, при этом берёт во внимание только тип элемента.
2. `:first-of-type` — позволяет выбрать первый дочерний элемент.
3. `:last-of-type` — выбирает последний тег конкретного типа.
4. `:nth-last-of-type()` — выбирает элемент заданного типа в списке элементов в соответствии с указанным местоположением начиная с конца.
5. `:only-of-type` — выбирает единственный элемент указанного типа среди дочерних элементов родительского элемента.

# Оформление результатов проверки

Хотя веб-разработчики не могут оформлять сообщения об ошибках проверки, они могут изменять внешний вид полей в зависимости от результатов их валидации. Например, можно выделить поле с неправильным значением цветным фоном сразу же, когда браузер обнаружит неправильные данные. Для этого требуется добавить несколько простых псевдоклассов

Доступны следующие опции:

Всё, что для этого требуется — это добавить несколько простых CSS3-псевдоклассов. Доступны следующие опции:

1. `required` и `optional` Применяют форматирование к полю в зависимости от того, использует ли это поле атрибут `required` или нет;
2. `valid` и `invalid` Применяют форматирование к полю в зависимости от правильности введенного в него значения. Но не забывайте, что большинство браузеров не проверяет данные, пока пользователь не попытается отправить форму, поэтому форматирование полей с некорректными значениями не выполняется сразу же при введении такого значения;
3. `in-range` и `out-of-range` Форматирование к полям, для которых используется атрибут `min` или `max`, чтобы ограничить их значение определенным диапазоном значений.

Пример кода в css

```
form input:required
{
    background-color: lightyellow;
}
```

## Пример использования псевдоклассов

HTML	CSS
<pre>&lt;ul&gt;   &lt;li&gt;Первый элемент&lt;/li&gt;   &lt;li&gt;Второй элемент&lt;/li&gt;   &lt;li&gt;Третий элемент&lt;/li&gt; &lt;/ul&gt;</pre>	<pre>li:first-child{   font-size: 24px;   color: #F23401; }</pre>

Чтобы понять, как работает этот псевдокласс (first-child), рассмотрим простой пример. Зададим элементу списка <li> псевдокласс first-child, и прописываем у него определённый стиль. Как видно, маркированный список состоит из трёх элементов. Указанный стиль применится ТОЛЬКО к первому элементу списка. Это происходит потому, что первый элемент списка <li> будет первым дочерним у тега <ul>.

HTML	CSS
<pre>&lt;ul&gt;   &lt;li&gt;&lt;a href="#"&gt;Первый элемент&lt;/a&gt;&lt;/li&gt;   &lt;li&gt;&lt;a href="#"&gt;Второй элемент&lt;/a&gt;&lt;/li&gt;   &lt;li&gt;&lt;a href="#"&gt;Третий элемент&lt;/a&gt;&lt;/li&gt; &lt;/ul&gt;</pre>	<pre>li:first-child a {   color: red; }</pre>

Одно из самых простых правил добавления псевдоклассов — просто проговорить, что требуется сделать.

Рассмотрим пример выше, есть список элементов, нам необходимо найти первый элемент списка и ссылку внутри него, проговариваем, найти первый элемент списка li, и внутри него ссылка, запись будет следующей li:first-child a { color: red; }

## Комбинирование псевдоклассов

Бывают такие ситуации, что нужно использовать сразу два псевдокласса, именно для таких ситуаций используется комбинирование.

Псевдоклассы комбинируются в одном селекторе и перечисляются через двоеточие:

```
/* При наведении на непосещенную ссылку цвет текста будет зеленым */
a:link:hover {
  color: #0F0;
}

/* При наведении на посещенную ссылку цвет текста будет красным*/
a:visited:hover {
  color: #F00;
}
```

# Псевдоэлементы

Псевдоэлементы позволяют ввести несуществующие элементы в веб-документ и придать им определённые стили. Псевдоэлементы появились ещё в CSS1, но пошли в релиз только в CSS2.1. В самом начале в синтаксисе использовалось одно двоеточие, но в CSS3 используется двойное двоеточие для отличия от псевдоклассов. Современные браузеры умеют понимать оба типа синтаксиса псевдоэлементов, кроме Internet Explorer 8. Он воспринимает только одно двоеточие. Поэтому надёжнее использовать одно. С помощью свойства `content` можно изменить внешний вид части элемента.

Рассмотрим часто используемые псевдоэлементы:

1. `:first-letter` — выбирает первую букву каждого абзаца, применяется только к блочным элементам.
2. `:first-line` — выбирает первую строку текста элемента, применяется только к блочным элементам.
3. `:before` — вставляет генерируемое содержимое перед элементом.
4. `:after` — добавляет генерируемое содержимое после элемента.

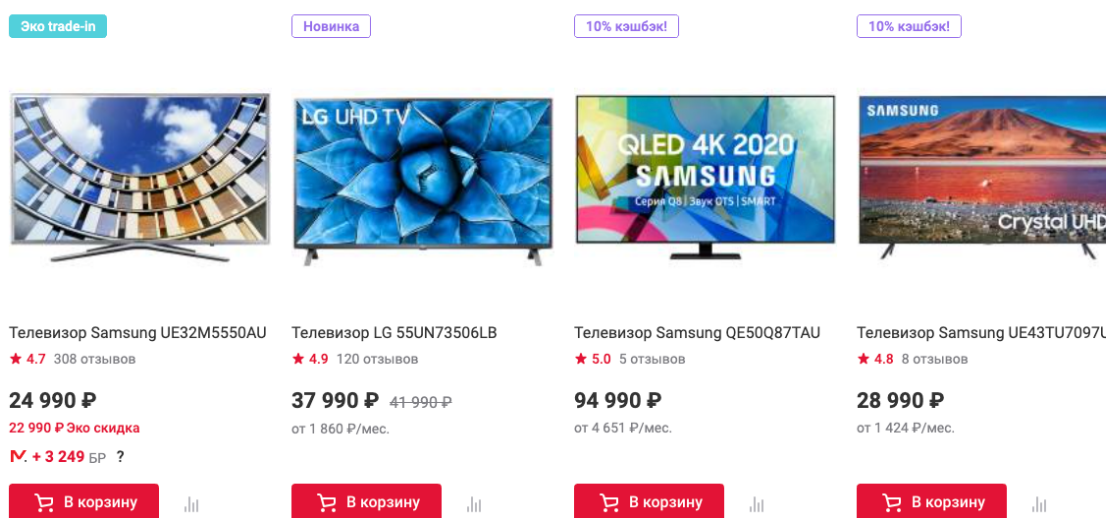
## Пример использования псевдоэлементов

HTML	CSS
<pre>&lt;ul&gt;   &lt;li&gt;Первый элемент&lt;/li&gt;   &lt;li&gt;Второй элемент&lt;/li&gt;   &lt;li&gt;Третий элемент&lt;/li&gt; &lt;/ul&gt;</pre>	<pre>li:after{   content: "new";   color: #F00; }</pre>

После всех элементов списка `li` появится текст `new` красного цвета.



## Пример использования псевдоэлементов на реальном сайте



На картинке шильдики «новинка», «10% кэшбэк!» и т. д. Эту часть проще всего сделать с помощью псевдоэлемента. Он не будет находиться в структуре html. Добавление подобного элемента состоит только из применения класса к любому блоку. Их можно легко поменять один на другой, не затрагивая html-файл.

Пример кода: <https://codepen.io/alexej-kadochnikow/pen/XWVNjVa?editors=1100>

## Выводы работы псевдоклассов и псевдоэлементов.

Давайте представим любой сайт в сети интернет, на какие элементы мы чаще всего наводим, конечно это ссылки и кнопки, но также стоит помнить про любые иконки, которые мы можем встретить на странице (соцсети и тд). Плюс для блоков мы можем добавлять не только текстовые элементы (10% кэшбек из предыдущего примера) но и изображения (иконки)

Теперь предлагаю вспомнить формы изображений, которые у нас есть, это: png, jpeg, gif, svg. Какой тут формат нас больше всего интересует, конечно же svg, так как у него может меняться значение цвета при наведении, может быть масштабирование, без потери качества, получается что данный формат отлично подходит для эффектов наведения, предлагаю более подробно разобраться с данным форматом.

## Использование SVG в HTML5

### Использование <object> тега

Если планируется использовать более продвинутые функции SVG, такие, как применение таблицы стилей CSS или внедрение скриптов, то тег HTML5 <object> один из вариантов.

```
<object type="image/svg+xml" data="image.svg" width="200" height="200" >  
Ваш браузер не поддерживает SVG  
</object>
```

Для старых браузеров, не поддерживающих SVG, можно использовать следующий метод:

```
<object type="image/svg+xml" data="SvgImg.svg" width="200" height="200">  
    
</object>
```

Браузер не понимающий SVG, проигнорирует тег <object> и перейдет к следующему тегу <img> и обработает его, как обычный HTML-тег и выведет картинку.

## Использование <iframe> тега

Так как браузеры могут отрисовывать по своим правилам SVG документы, то это дает возможность загружать картинки внутри тегов <iframe>.

```
<iframe src="SvgImg.svg">  
    
</iframe>
```

Это может быть хорошим методом, если вы хотите полностью отделить SVG-код и скрипт на вашей главной странице.

## Использование <img> тега

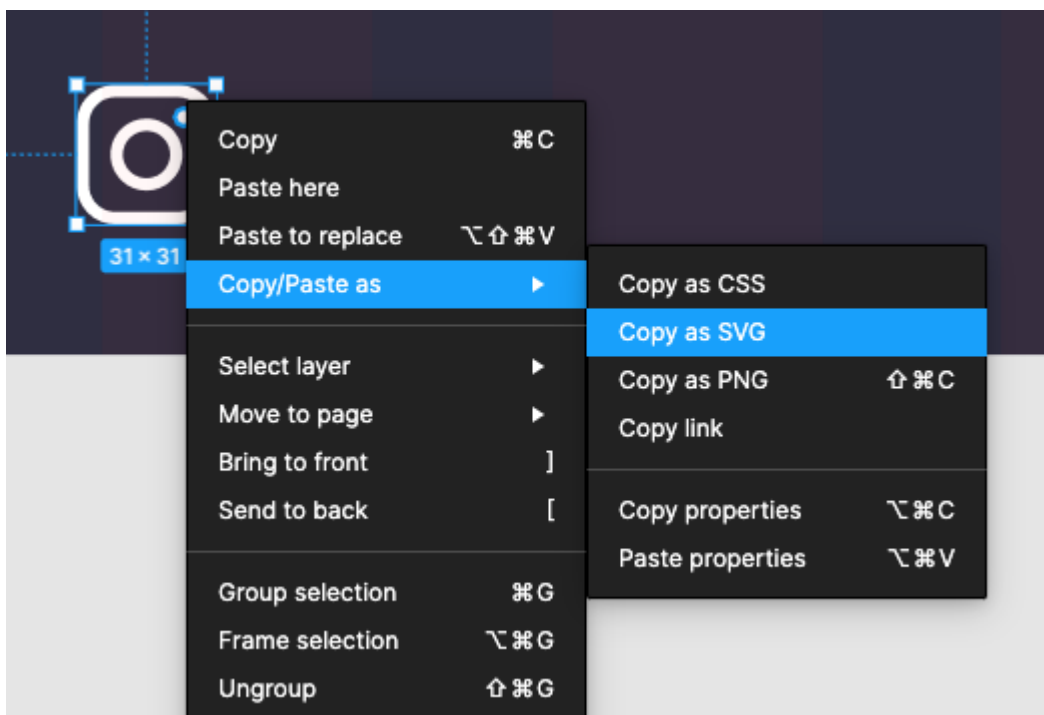
SVG-документ может быть добавлен на вашу веб страницу, как любое другое изображение:

```

```

## Использование <svg> тега

Одним из самых простых способов добавления иконки в формате svg это как раз использовать тег <svg>



При нажатии на любое изображение в векторном формате, вы можете скопировать код svg, для этого вам необходимо нажать на него правой клавишей и выбрать “Copy as SVG” . Мы как обычно работаем в программе Figma.

## Использование CSS Background Image

SVG может быть использован в качестве CSS фона для любого элемента:

```
#AnyElement
{
  background-image: url(image.svg);
}
```

Как и при использовании тега `<img>` связывание, скриптование и другие методы интерактивности будут недоступны.

# Как добавить эффект наведения на svg изображение

Чтобы добавить эффект наведения на нужную нам svg картинку вам потребуется добавить ее в формате svg в код html и далее создать эффект наведения на этот элемент path, для этого вам потребуется поменять свойство **fill** для данной картинки.

html	css
<pre>&lt;svg class="icon" width="9" height="15" viewBox="0 0 9 15" fill="none" xmlns="http://www.w3.org/2000/svg"&gt; &lt;path d="M8.08836 8.28L8.50686 5.61602H5.89022V3.88729C5.89022 3.15847 6.25574 2.44806 7.42765 2.44806H8.61722V0.179975C8.61722 0.179975 7.53772 0 6.50561 0C4.35073 0 2.9422 1.27593 2.9422 3.5857V5.61602H0.546875V8.28H2.9422V14.72H5.89022V 8.28H8.08836Z" fill="black"/&gt; &lt;/svg&gt;</pre>	<pre>.icon:hover path {   fill: red; }</pre>

## CSS3 Трансформация

Что мы уже знаем про блочные элементы, это то что они отображаются на веб-странице в виде прямоугольника, но как быть в ситуациях если нам необходимо добавить наклон блока или может быть нам потребуется создания треугольника, именно в этот момент к нам на помощь приходит трансформация элемента.

### transform

Свойство задаёт вид преобразования элемента. Свойство описывается с помощью функций трансформации, которые смещают элемент относительно его текущего положения на странице или изменяют его первоначальные размеры и форму. Не наследуется.

Допустимые значения:

- `matrix()` — любое число;
- `translate()`, `translateX()`, `translateY()` — единицы длины (положительные и отрицательные), %;
- `scale()`, `scaleX()`, `scaleY()` — любое число;
- `rotate()` — угол (deg, grad, rad или turn);
- `skew()`, `skewX()`, `skewY()` — угол (deg, grad, rad).

1. `matrix(a, c, b, d, x, y)`

- Смещает элементы и задает способ их трансформации, позволяя объединить несколько функций 2D-трансформаций в одной. В качестве трансформации допустимы поворот, масштабирование, наклон и изменение положения. Значение `a` изменяет масштаб по горизонтали.
- Значение от 0 до 1 уменьшает элемент, больше 1 — увеличивает.
- Значение `c` деформирует (сдвигает) стороны элемента по оси `Y`, положительное значение — вверх, отрицательное — вниз.
- Значение `b` деформирует (сдвигает) стороны элемента по оси `X`, положительное значение — влево, отрицательное — вправо.
- Значение `d` изменяет масштаб по вертикали. Значение меньше 1 уменьшает элемент, больше 1 — увеличивает.
- Значение `x` смещает элемент по оси `X`, положительное — вправо, отрицательное — влево.
- Значение `y` смещает элемент по оси `Y`, положительное значение — вниз, отрицательное — вверх.

2. `translate(x,y)`

- Сдвигает элемент на новое место, перемещая относительно обычного положения вправо и вниз, используя координаты `x` и `y`, не затрагивая при этом соседние элементы. Если нужно сдвинуть элемент влево или вверх, то нужно использовать отрицательные значения.

3. `translateX(n)`

- Сдвигает элемент относительно его обычного положения по оси `X`.

4. `translateY(n)`

- Сдвигает элемент относительно его обычного положения по оси `Y`.

5. `scale(x,y)`

- Масштабирует элементы, делая их больше или меньше. Значения от 0 до 1 уменьшают элемент. Первое значение масштабирует элемент по ширине, второе — по высоте. Отрицательные значения отображают элемент зеркально.
- `scaleX(n)` Функция масштабирует элемент по ширине, делая его шире или уже. Если значение больше единицы, элемент становится шире, если значение находится между единицей и нулем, элемент становится

уже. Отрицательные значения отображают элемент зеркально по горизонтали.

- c. `scaleY(n)` Функция масштабирует элемент по высоте, делая его выше или ниже. Если значение больше единицы, элемент становится выше, если значение находится между единицей и нулем — ниже. Отрицательные значения отображают элемент зеркально по вертикали.

#### 6. `rotate(угол)`

- a. Поворачивает элементы на заданное количество градусов, отрицательные значения от `-1deg` до `-360deg` поворачивают элемент против часовой стрелки, положительные — по часовой стрелке. Значение `rotate(720deg)` поворачивает элемент на два полных оборота.

#### 7. `skew(x-угол,y-угол)`

- a. Используется для деформирования (искажения) сторон элемента относительно координатных осей. Если указано одно значение, второе будет определено браузером автоматически.
- b. `skewX(угол)` Деформирует стороны элемента относительно оси X.
- c. `skewY(угол)` Деформирует стороны элемента относительно оси Y.
- d. `initial` Устанавливает значение свойства в значение по умолчанию.
- e. `inherit` Наследует значение свойства от родительского элемента.

## CSS3 transition

При добавлении плавности переходов для элементов, для общего свойства `transition` существуют его составляющие (аналогично свойству `background`, `background-color`, `background-image` и т.д)

Рассмотрим из чего состоит общее свойство `transition`

1. Свойство `transition-property`. Содержит название CSS-свойств, к которым будет применен эффект перехода. Не наследуется.
  - a. Значения.
    - i. **none** - отсутствие свойства для перехода.
    - ii. **all** - значение по умолчанию. Применяет эффект перехода ко всем свойствам элемента.
    - iii. **свойство** - определяет список css-свойств, перечисленных через запятую, участвующих в переходе.

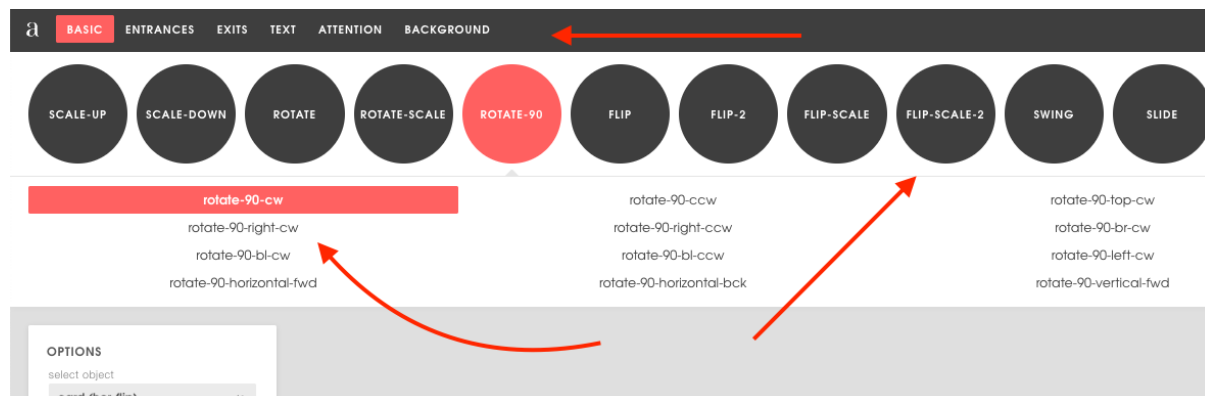
2. Продолжительность перехода `transition-duration` Задаёт промежуток времени, в течение которого должен осуществляться переход. Не наследуется.
  - а. Значения.
    - i. **время** перехода указывается в секундах или миллисекундах, например, 1s или 5ms. или 0.3s
3. Функция перехода `transition-timing-function`. Свойство задаёт временную функцию, которая определяет скорость перехода объекта от одного значения к другому. Если вы определяете более одного перехода для элемент, например, цвет фона элемента и его положение, вы
  - а. Значения
    - i. **ease** — функция по умолчанию, переход начинается медленно, разгоняется быстро и замедляется в конце. Соответствует `cubic-bezier(0.25,0.1,0.25,1)`.
    - ii. **linear** — переход происходит равномерно на протяжении всего времени, без колебаний в скорости. Соответствует `cubic-bezier(0,0,1,1)`.
    - iii. **ease-in** — переход начинается медленно, а затем плавно ускоряется в конце. Соответствует `cubic-bezier(0.42,0,1,1)`.
    - iv. **ease-out** — переход начинается быстро и плавно замедляется в конце. Соответствует `cubic-bezier(0,0,0.58,1)`.
    - v. **ease-in-out** — переход медленно начинается и медленно заканчивается. Соответствует `cubic-bezier(0.42,0,0.58,1)`.
    - vi. **cubic-bezier(x1, y1, x2, y2)** — позволяет вручную установить значения от 0 до 1 для кривой ускорения.
4. Задержка перехода `transition-delay`. Необязательное свойство, позволяет сделать так, чтобы изменение свойства происходило не моментально, а с некоторой задержкой. Не наследуется.
  - а. **Время** задержки перехода указывается в секундах или миллисекундах.
5. Краткая запись перехода `transition`. Все свойства, отвечающие за изменение внешнего вида элемента, можно объединить в одно свойство `transition`

## CSS3 Анимация

Перед тем, как мы начнём разбираться, как добавить анимацию на странице, давайте посмотрим какие анимации мы можем встретить на странице <https://animista.net/play/basic/rotate-scale>



Выбираем любой раздел и смотрим примеры анимаций



Для создания анимации в CSS3 используется свойство @keyframes.

Данное свойство представляет собой контейнер, в который должны помещаться различные свойства оформления.

```
@keyframes имяАнимации
{
  from {CSS свойства} /* Оформление элемента перед началом анимации */
  to {CSS свойства} /* Оформление элемента после завершения анимации */
}
```

После того, как анимация была создана, необходимо добавить к элементу, который Вы хотите анимировать, CSS3 свойство animation и указать в нём имя анимации (1 значение) и время (2 значение), в течение которого она будет выполняться.

Также Вы можете устанавливать количество повторов анимации (3 значение).

```
@keyframes anim {
  from {margin-left:3px;}
  to {margin-left:500px;}
}
#wrap1 {
  animation:anim 4s 3;
}
```

# Длительность анимации animation-duration

Свойство устанавливает длительность анимации. Не наследуется. Значение по умолчанию 0.

Значения	Описание
время	Длительность анимации задается в секундах s или миллисекундах ms.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
-webkit-animation-duration: 2s;  
animation-duration: 2s;
```

## Временная функция animation-timing-function

Свойство определяет изменение скорости от начала до конца анимации с помощью временных функций.

Значения	Описание
ease	Функция по умолчанию, анимация начинается медленно, разгоняется быстро и замедляется в конце. Соответствует cubic-bezier(0.25,0.1,0.25,1).
linear	Анимация происходит равномерно на протяжении всего времени, без колебаний в скорости. Соответствует cubic-bezier(0,0,1,1).
ease-in	Анимация начинается медленно, а затем плавно ускоряется в конце. Соответствует cubic-bezier(0.42,0,1,1).
ease-out	Анимация начинается быстро и плавно замедляется в конце. Соответствует cubic-bezier(0,0,0.58,1).
ease-in-out	Анимация медленно начинается и медленно заканчивается. Соответствует cubic-bezier(0.42,0,0.58,1).
cubic-bezier(x1, y1, x2, y2)	Позволяет вручную установить значения от 0 до 1. <a href="#">На этом сайте</a> вы сможете построить любую траекторию скорости изменения анимации.
step-start	Задаёт пошаговую анимацию, разбивая анимацию на отрезки, изменения происходят в начале каждого шага. Эквивалентно steps(1, start).
step-end	Пошаговая анимация, изменения происходят в конце каждого шага. Эквивалентно steps(1, end).
steps(количество шагов,start end)	Ступенчатая временная функция, которая принимает два параметра. Количество шагов задается целым положительным числом. Второй параметр необязательный, указывает момент, в котором начинается анимация. 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.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

## Шаги анимации

Элемент можно анимировать используя шаги, т.е. ступенчато. (Примером может послужить секундная стрелка часов, которая сначала двигается, а затем

осуществляется задержка на 1 секунду, потом снова двигается и снова задержка и т.д.) Шаги задаются с помощью функции `steps()`. Ниже показан пример:

```
transition: all 2000ms steps(3, end);
```

Теперь анимация увеличения кнопки будет происходить рывками. Задержка между рывками в данном случае будет 667 ms (2000/3). Второй параметр функции `steps` указывает, будет ли рывок выполняться сразу или после задержки.

## Анимация с задержкой `animation-delay`

Свойство игнорирует анимацию заданное количество времени, что даёт возможность по отдельности запускать каждую анимацию. Отрицательная задержка начинает анимацию с определенного момента внутри её цикла, т.е. со времени, указанного в задержке. Это позволяет применять анимацию к нескольким элементам со сдвигом фазы, изменяя лишь время задержки.

Чтобы анимация началась с середины, нужно задать отрицательную задержку, равную половине времени, установленном в `animation-duration`. Не наследуется.

Значения	Описание
время	Задержка анимации задается в секундах s или миллисекундах ms. Значение по умолчанию 0.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис:

```
-webkit-animation-delay: 2s;  
animation-delay: 2s;
```

## Повтор анимации animation-iteration-count

Свойство позволяет запустить анимацию несколько раз. Значение 0 или любое отрицательное число удаляют анимацию из проигрывания. Не наследуется.

Значения	Описание
число	С помощью целого числа задается количество повторов анимации. Значение по умолчанию 1.
infinite	Анимация проигрывается бесконечно.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис:

```
-webkit-animation-iteration-count: 3;  
animation-iteration-count: 3;
```

## Направление анимации animation-direction

Свойство задает направление повтора анимации. Если анимация повторяется только один раз, то это свойство не имеет смысла. Не наследуется.

Значения	Описание
alternate	Анимация проигрывается с начала до конца, затем в обратном направлении.
alternate-reverse	Анимация проигрывается с конца до начала, затем в обратном направлении.
normal	Значение по умолчанию, анимация проигрывается в обычном направлении, с начала и до конца.
reverse	Анимация проигрывается с конца.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
-webkit-animation-direction: alternate;
animation-direction: alternate;
```

## Проигрывание анимации

### animation-play-state

Свойство управляет проигрыванием и остановкой анимации. Остановка анимации внутри цикла возможна через использование этого свойства в скрипте JavaScript. Также можно останавливать анимацию при наведении курсора мыши на объект — состояние `:hover`. Не наследуется.

Значения	Описание
paused	Останавливает анимацию.
running	Значение по умолчанию, означает проигрывание анимации.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис:

```
-webkit-animation-play-state: paused;
animation-play-state: paused;
```

## Состояние элемента до и после воспроизведения анимации

### animation-fill-mode

Свойство определяет порядок применения определенных в `@keyframes` стилей к объекту. Не наследуется.

Значения	Описание
none	Значение по умолчанию. Состояние элемента не меняется до или после воспроизведения анимации.
forwards	Воспроизводит анимацию до последнего кадра по окончании последнего повтора и не отматывает ее к первоначальному состоянию.
backwards	Возвращает состояние элемента после загрузки страницы к первому кадру, даже если установлена задержка animation-delay, и оставляет его там, пока не начнется анимация.
both	Позволяет оставлять элемент в первом ключевом кадре до начала анимации (игнорируя положительное значение задержки) и задерживать на последнем кадре до конца последней анимации.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис:

```
-webkit-animation-fill-mode: forwards;
animation-fill-mode: forwards;
```

## Краткая запись анимации

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;
```

## Множественные анимации

Для одного элемента можно задавать несколько анимаций, перечислив их названия через запятую:

Синтаксис:

```
div {animation: shadow 1s ease-in-out 0.5s alternate, move 5s linear 2s;}
```