

Раздел

[Анимация](#)

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие transitionend

Ключевые кадры

Итого


Задачи (3)

Комментарии

Поделиться



Редактировать на GitHub

 → [Анимация](#) 14-го августа 2019

CSS-анимации

CSS позволяет создавать простые анимации без использования JavaScript.

JavaScript может быть использован для управления такими CSS-анимациями. Это позволяет делать более сложные анимации, используя небольшие кусочки кода.

CSS-переходы

Идея CSS-переходов проста: мы указываем, что некоторое свойство должно быть анимировано, и как оно должно быть анимировано. А когда свойство меняется, браузер сам обработает это изменение и отрисует анимацию.

Всё что нам нужно, чтобы начать анимацию – это изменить свойство, а дальше браузер сделает плавный переход сам.

Например, CSS-код ниже анимирует трёх-секундное изменение background-color :

```
1 .animated {
2   transition-property: background-color;
3   transition-duration: 3s;
4 }
```

Теперь, если элементу присвоен класс .animated, любое изменение свойства background-color будет анимироваться в течение трёх секунд.

Нажмите кнопку ниже, чтобы анимировать фон:

```
1 <button id="color">Нажми меня</button>
2
3 <style>
4   #color {
5     transition-property: background-color;
6     transition-duration: 3s;
7   }
8 </style>
9
10 <script>
11   color.onclick = function() {
12     this.style.backgroundColor = 'red';
13   };
14 </script>
```



Существует 4 свойства для описания CSS-переходов:

- transition-property – свойство перехода
- transition-duration – продолжительность перехода
- transition-timing-function – временная функция перехода
- transition-delay – задержка начала перехода

Далее мы рассмотрим их все, а сейчас ещё заметим, что есть также общее свойство transition, которое позволяет задать их одновременно в последовательности: property duration timing-function delay, а также анимировать несколько свойств одновременно.

Например, у этой кнопки анимируются два свойства color и font-size одновременно:

Раздел

Анимация

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие transitionend

Ключевые кадры

Итого

Задачи (3)

Комментарии

Поделиться



Редактировать на GitHub



```
1 <button id="growing">Нажми меня</button>
2
3 <style>
4 #growing {
5     transition: font-size 3s, color 2s;
6 }
7 </style>
8
9 <script>
10 growing.onclick = function() {
11     this.style.fontSize = '36px';
12     this.style.color = 'red';
13 };
14 </script>
```

Нажми меня

Теперь рассмотрим каждое свойство анимации по отдельности.

transition-property

В transition-property записывается список свойств, изменения которых необходимо анимировать, например: left, margin-left, height, color.

Анимировать можно не все свойства, но [многие из них](#). Значение свойства all означает «анимируй все свойства».

transition-duration

В transition-duration можно определить, сколько времени займёт анимация. Время должно быть задано в [формате времени CSS](#): в секундах s или миллисекундах ms.

transition-delay

В transition-delay можно определить задержку *перед* началом анимации. Например, если transition-delay: 1s, тогда анимация начнётся через 1 секунду после изменения свойства.

Отрицательные значения также допустимы. В таком случае анимация начнётся с середины. Например, если transition-duration равно 2s, а transition-delay – -1s, тогда анимация займёт одну секунду и начнётся с середины.

Здесь приведён пример анимации, сдвигающей цифры от 0 до 9 с использованием CSS-свойства transform со значением translate:

Результат script.js style.css index.html

Кликните на цифру для начала анимации:

0

Свойство transform анимируется следующим образом:

```
1 #stripe.animate {
2     transform: translate(-90%);
3     transition-property: transform;
4     transition-duration: 9s;
5 }
```

В примере выше JavaScript-код добавляет класс `.animate` к элементу, после чего начинается анимация:

```
1 stripe.classList.add('animate');
```

Можно начать анимацию «с середины», с определённого числа, например, используя отрицательное значение `transition-delay`, соответствующие необходимому числу.

Если вы нажмёте на цифру ниже, то анимация начнётся с последней секунды:

Результат script.js style.css index.html

Click below to animate:

0

JavaScript делает это с помощью нескольких строк кода:

```
1 stripe.onclick = function() {
2   let sec = new Date().getSeconds() % 10;
3   // например, значение -3s здесь начнут анимацию с тре
4   stripe.style.transitionDelay = '-' + sec + 's';
5   stripe.classList.add('animate');
6 };
```

transition-timing-function

Временная функция описывает, как процесс анимации будет распределён во времени. Будет ли она начата медленно и затем ускорится или наоборот.

На первый взгляд это очень сложное свойство, но оно становится понятным, если уделить ему немного времени.

Это свойство может принимать два вида значений: кривую Безье или количество шагов. Давайте начнём с кривой Безье, как с наиболее часто используемой.

Кривая Безье

Временная функция может быть задана, как [кривая Безье](#) с 4 контрольными точками, удовлетворяющими условиям:

1. Первая контрольная точка: $(0, 0)$.
2. Последняя контрольная точка: $(1, 1)$.
3. Для промежуточных точек значение x должно быть $0..1$, значение y может принимать любое значение.

Синтаксис для кривых Безье в CSS: `cubic-bezier(x2, y2, x3, y3)`. Нам необходимо задать только вторую и третью контрольные точки, потому что первая зафиксирована со значением $(0, 0)$ и четвёртая – $(1, 1)$.

Временная функция описывает то, насколько быстро происходит анимации во времени.

- Ось x – это время: 0 – начальный момент, 1 – последний момент `transition-duration`.
- Ось y указывает на завершение процесса: 0 – начальное значение свойства, 1 – конечное значение.

Самым простым примером анимации является равномерная анимация с линейной скоростью. Она может быть задана с помощью кривой `cubic-`

Раздел

[Анимация](#)

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие `transitionend`

Ключевые кадры

Итого

Задачи (3)

Комментарии

Поделиться



Редактировать на GitHub

Раздел

Анимация

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие transitionend

Ключевые кадры

Итого

Задачи (3)

Комментарии

Поделиться



Редактировать на GitHub



`bezier(0, 0, 1, 1)`.

Вот как выглядит эта «кривая»:



...Как мы видим, это прямая линия. Значению времени (x) соответствует значение завершенности анимации (y), которое равномерно изменяется от 0 к 1.

В примере ниже поезд «едет» слева направо с одинаковой скоростью (нажмите на поезд):

Результат style.css index.html



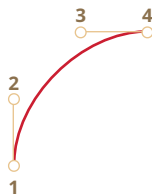
В свойстве `transition` указана следующая кривая Безье:

```
1 .train {  
2   left: 0;  
3   transition: left 5s cubic-bezier(0, 0, 1, 1);  
4   /* JavaScript устанавливает свойство left равным 450p  
5 }
```

...А как показать замедляющийся поезд?

Мы можем использовать другую кривую Безье: `cubic-bezier(0.0, 0.5, 0.5, 1.0)`.

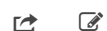
Её график:



Как видим, анимация начинается быстро: кривая быстро поднимается вверх, и затем все медленнее и медленнее.

Ниже временная функция в действии (нажмите на поезд):

Результат style.css index.html



CSS:

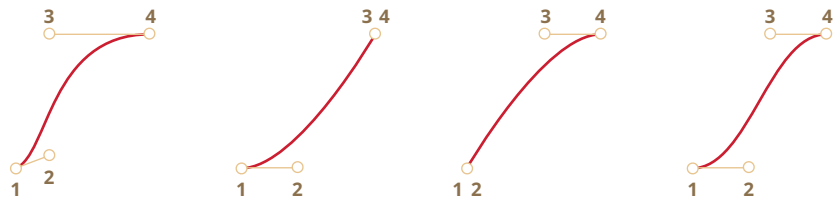
```
1 .train {
2   left: 0;
3   transition: left 5s cubic-bezier(0, .5, .5, 1);
4   /* JavaScript устанавливает свойство left равным 450px
5 }
```

Есть несколько встроенных обозначений кривых Безье: `linear`, `ease`, `ease-in`, `ease-out` и `ease-in-out`.

`linear` это короткая запись для `cubic-bezier(0, 0, 1, 1)` – прямой линии, которую мы видели раньше.

Другие названия – это также сокращения для других `cubic-bezier`:

ease *	ease-in	ease-out	ease-in-out
(0.25, 0.1, 0.25, 1.0)	(0.42, 0, 1.0, 1.0)	(0, 0, 0.58, 1.0)	(0.42, 0, 0.58, 1.0)



* – используется по умолчанию, если не задана другая временная функция.

Для того, чтобы замедлить поезд, мы можем использовать `ease-out`:

```
1 .train {
2   left: 0;
3   transition: left 5s ease-out;
4   /* transition: left 5s cubic-bezier(0, .5, .5, 1); */
5 }
```

Но получившийся результат немного отличается.

Кривая Безье может заставить анимацию «выпрыгивать» за пределы диапазона.

Контрольные точки могут иметь любые значения по оси `y`: отрицательные или сколь угодно большие. В таком случае кривая Безье будет скакать очень высоко или очень низко, заставляя анимацию выходить за её нормальные пределы.

В приведённом ниже примере код анимации:

```
1 .train {
2   left: 100px;
3   transition: left 5s cubic-bezier(.5, -1, .5, 2);
4   /* JavaScript sets left to 400px */
5 }
```

Свойство `left` будет анимироваться от `100px` до `400px`.

Но когда вы нажмёте на поезд, вы увидите следующее:

- Сначала, поезд поедет *назад*: `left` станет меньше, чем `100px`.
- Затем он поедет *вперёд*, немного дальше, чем `400px`.
- И затем вернётся назад в значение `400px`.

Раздел

Анимация

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие transitionend

Ключевые кадры

Итого

Задачи (3)

Комментарии

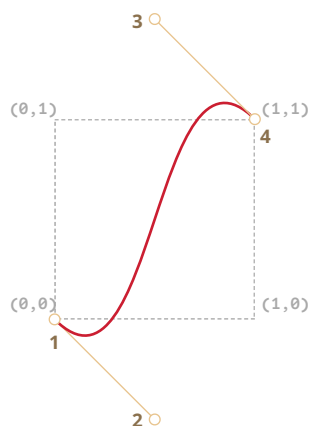
Поделиться



Редактировать на GitHub



Если мы взглянем на кривую Безье из примера, становится понятно поведение поезда.



Мы вынесли координату y для первой опорной точки ниже нуля и выше единицы для третьей опорной точки, поэтому кривая вышла за пределы «обычного» квадрата. Значения y вышли из «стандартного» диапазона $0 \dots 1$.

Как мы знаем, ось y измеряет «завершённость процесса анимации». Значение $y = 0$ соответствует начальному значению анимируемого свойства и $y = 1$ – конечному значению. Таким образом, $y < 0$ делает значение свойства `left` меньше начального значения и $y > 1$ – больше конечного.

Это, конечно, «мягкий» вариант. Если значение y будут -99 и 99 , то поезд будет гораздо сильнее «выпрыгивать» за пределы.

Как сделать кривую Безье необходимую для конкретной задачи? Существует множество инструментов, например можно использовать с сайта <http://cubic-bezier.com/>.

Шаги

Временная функция `steps(количество шаров[, start/end])` позволяет разделить анимацию на шаги.

Давайте рассмотрим это на уже знакомом нам примере с цифрами.

Ниже представлен список цифр, без какой-либо анимации, который мы будем использовать в качестве основы:

Результат style.css index.html



0123456789

Раздел

Анимация

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие transitionend

Ключевые кадры

Итого

Задачи (3)

Комментарии

Поделиться



Редактировать на GitHub



Давайте сделаем так, чтобы цифры двигались не плавно, а появлялись одна за другой отдельно. Для этого скроем все что находится за красным «окошком» и будем сдвигать список влево по шагам.

Всего будет 9 шагов, один шаг для каждой цифры:

```
1 #stripe.animate {
2   transform: translate(-90%);
3   transition: transform 9s steps(9, start);
4 }
```

В действии:

Результат style.css index.html



Кликните на цифру для начала анимации:

0

Первый аргумент временной функции `steps(9, start)` – количество шагов. Трансформация будет разделена на 9 частей (10% каждая). Временной интервал также будет разделён на 9 частей, таким образом свойство `transition: 9s` обеспечивает нам 9 секунду анимации, что даёт по одной секунде на цифру.

Вторым аргументом является одно из ключевых слов: `start` или `end`.

`start` – означает, что в начале анимации нам необходимо перейти на первый шаг немедленно.

Мы можем наблюдать это во время анимации: когда пользователь нажимает на цифру, значение меняется на 1 (первый шаг) сразу и в следующий раз меняется уже в начале следующей секунды.

Анимация будет происходить так:

- 0s – -10% (первое изменение в начале первой секунды, сразу после нажатия)
- 1s – -20%
- ...
- 8s – -80%
- (на протяжении последней секунды отображается последнее значение).

Альтернативное значение `end` означало бы, что изменения нужно применять не в начале, а в конце каждой секунды.

Анимация будет происходить так:

- 0s – 0
- 1s – -10% (первое изменение произойдёт в конце первой секунды)
- 2s – -20%
- ...
- 9s – -90%

Пример `step(9, end)` в действии (обратите внимание на паузу между первым изменением цифр):

Результат style.css index.html



Раздел

[Анимация](#)

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие transitionend

Ключевые кадры

Итого

Задачи (3)

Комментарии

Поделиться



Редактировать на GitHub



Кликните на цифру для начала анимации:

0

Также есть сокращённые значения:

- `step-start` – то же самое, что `steps(1, start)`. Оно означает, что анимация начнётся сразу и произойдёт в один шаг. Таким образом она начнётся и завершится сразу, как будто и нет никакой анимации.
- `step-end` – то же самое, что `steps(1, end)`: выполнит анимацию за один шаг в конце `transition-duration`.

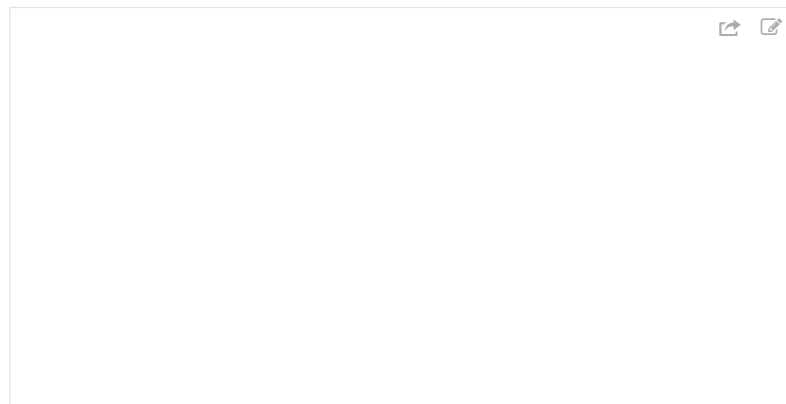
Такие значения используются редко, потому что это не совсем анимация, а точнее будет сказать одношаговые изменения.

Событие transitionend

Когда завершается анимация, срабатывает событие `transitionend`.

Оно широко используется для выполнения действий после завершения анимации, а также для создания последовательности анимаций.

Например, корабль в приведённом ниже примере начинает плавать туда и обратно по клику, каждый раз все дальше и дальше вправо:



Анимация начинается с помощью функции `go`, которая вызывается каждый раз снова, когда переход заканчивается и меняется направление:

```
1 boat.onclick = function() {
2   //...
3   let times = 1;
4
5   function go() {
6     if (times % 2) {
7       // плыть вправо
8       boat.classList.remove('back');
9       boat.style.marginLeft = 100 * times + 200 + 'px';
10    } else {
11      // плыть влево
12      boat.classList.add('back');
13      boat.style.marginLeft = 100 * times - 200 + 'px';
14    }
15  }
16
17  go();
18
19  boat.addEventListener('transitionend', function() {
20    times++;
21    go();
22  });
23
24  };
```


Раздел

[Анимация](#)

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие transitionend

Ключевые кадры

Итого

Задачи (3)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Объект события `transitionend` содержит ряд полезных свойства:

`event.propertyName`

Имя свойства, анимация которого завершилась. Может быть полезным, если мы анимируем несколько свойств.

`event.elapsedTime`

Время (в секундах), которое заняла анимация, без учёта `transition-delay`.

Ключевые кадры

Мы можем объединить несколько простых анимаций вместе, используя CSS-правило `@keyframes`.

Оно определяет «имя» анимации и правила: что, когда и где анимировать. После этого можно использовать свойство `animation`, чтобы назначить анимацию на элемент и определить её дополнительные параметры.

Ниже приведён пример с пояснениями:

```
1 <div class="progress"></div>
2
3 <style>
4   @keyframes go-left-right {           /* объявляем имя ан
5     from { left: 0px; }                 /* от: left: 0px */
6     to { left: calc(100% - 50px); }     /* до: left: 100%-50px */
7   }
8
9   .progress {
10    animation: go-left-right 3s infinite alternate;
11    /* применить анимацию "go-left-right" на элементе
12       продолжительностью 3 секунды
13       количество раз: бесконечно (infinite)
14       менять направление анимации каждый раз (alternate) */
15
16
17    position: relative;
18    border: 2px solid green;
19    width: 50px;
20    height: 20px;
21    background: lime;
22  }
23 </style>
```



Существует множество статей про `@keyframes`, а также [детальная спецификация](#).

Скорее всего, вам нечасто понадобится `@keyframes`, разве что на вашем сайте все постоянно в движении.

Итого

CSS-анимации позволяют плавно, или не очень, менять одно или несколько свойств.

Они хорошо решают большинство задач по анимации. Также мы можем реализовать анимации через JavaScript, более подробно об этом – в следующей главе.

Ограничения CSS-анимаций в сравнении с JavaScript-анимациями:

Достоинства

Недостатки

Раздел

[Анимация](#)

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие transitionend

Ключевые кадры

Итого

Задачи (3)

Комментарии

Поделиться



[Редактировать на GitHub](#)



- Простые анимации делаются просто.
- Быстрые и не создают нагрузку на CPU.

- JavaScript-анимации более гибкие. В них может присутствовать любая анимационная логика, как например «взорвать» элемент.
- Можно изменять не только свойства. Мы можем создавать новые элементы с помощью JavaScript для анимации.

Большинство анимаций может быть реализовано с использованием CSS, как описано в этой главе. А событие `transitionend` позволяет запускать JavaScript после анимации, поэтому CSS-анимации прекрасно интегрируются с кодом.

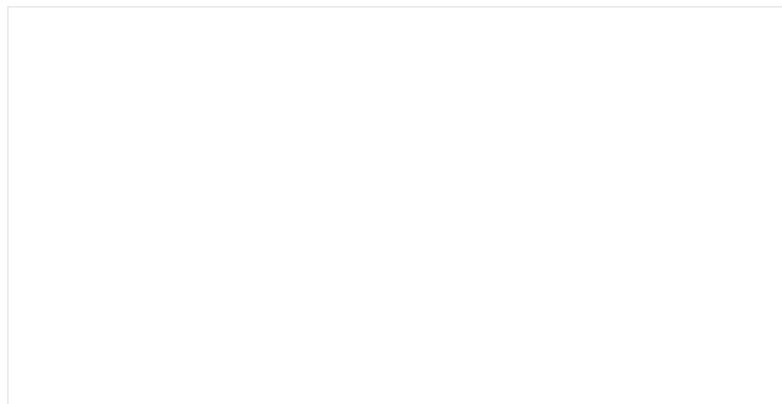
Но в следующей главе мы рассмотрим некоторые JavaScript-анимации, которые позволяют решать более сложные задачи.

✓ Задачи

Анимировать самолёт (CSS)

важность: 5

Реализуйте анимацию, как в примере ниже (клик на самолёт):



- При нажатии картинка изменяет размеры с 40x24px до 400x240px (увеличивается в 10 раз).
- Время анимации 3 секунды.
- По окончании анимации вывести сообщение: «Анимация закончилась!».
- Если во время анимации будут дополнительные клики по картинке – они не должны ничего «сломать».

[Открыть песочницу для задачи.](#)

решение

Анимировать самолёт с перелётом (CSS)

важность: 5

Модифицируйте решение предыдущей задачи [Анимировать самолёт \(CSS\)](#), чтобы в процессе анимации изображение выросло больше своего стандартного размера 400x240px («выпрыгнуло»), а затем вернулось к нему.

Должно получиться, как в примере ниже (клик на самолёт):

Раздел

[Анимация](#)

Навигация по уроку

CSS-переходы

transition-property

transition-duration

transition-delay

transition-timing-function

Событие transitionend

Ключевые кадры

Итого

Задачи (3)

Комментарии

Поделиться



Редактировать на GitHub



В качестве исходного кода возьмите решение прошлой задачи.

[решение](#)

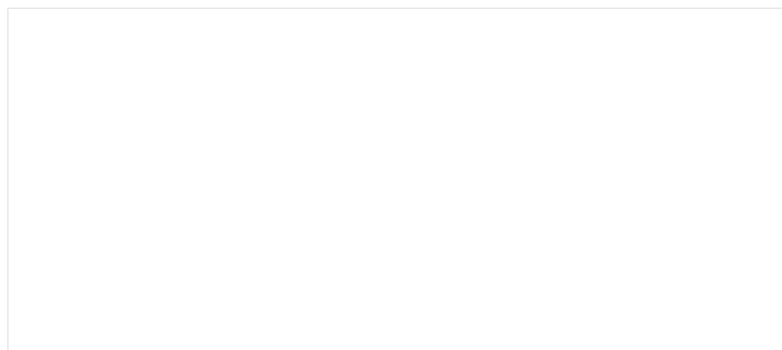
Анимированный круг [↗](#)

важность: 5

Напишите функцию `showCircle(cx, cy, radius)`, которая будет рисовать постепенно растущий круг.

- `cx, cy` – координаты центра круга относительно окна браузера,
- `radius` – радиус круга.

Нажмите на кнопку ниже, чтобы увидеть как это должно выглядеть:



В исходном коде уже указаны правильные CSS-стили круга, таким образом задача заключается в том, чтобы сделать правильную анимацию.

[Открыть песочницу для задачи.](#)

[решение](#)

Проводим [курсы по JavaScript и фреймворкам.](#)



💬 Комментарии

перед тем как писать...