# Преобразования

Используется свойство **transform:**



Тип трансформации иногда называют еще **функцией трансформации**

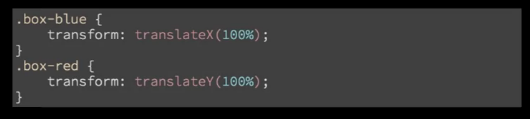
Важно то, что при трансформации физическое местонахождение объекта не меняется. То есть перемещение объекта не будет влиять на окружающие объекты никак.

#### Перемещение

Функция - **translate(x,y)** - аргументы - перемещение по оси Х(вправо) и по оси Y(вниз).

Аргументы также могут задаваться в процентах от размера перемещаемого блока.

Указание перемещения только по одной оси:

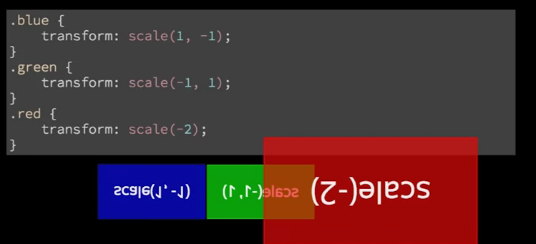


#### Масштабирование

Функция **scale(x).**

Значение аргумента - без единиц измерения 0.5 - уменьшить в два раза. Могут быть указаны два аргумента - тогда изменение размера будет проходить отдельно для оси Х и для оси Y. Также есть функции **scaleX** и **scaleY**, которые работают аналогично вышеупомянутым для перемещения.

Также можно указывать отрицательные значения, тогда будет происходить переворот элемента.



Элемент сначала сожмется, потом перевернется и расширится.

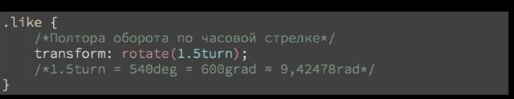
#### Вращение

Функция **rotate(180deg)**

**Deg** - градусы

Положительное значение - по часовой стрелке

Также можно указывать величину в радианах, градусах и количестве поворотов.



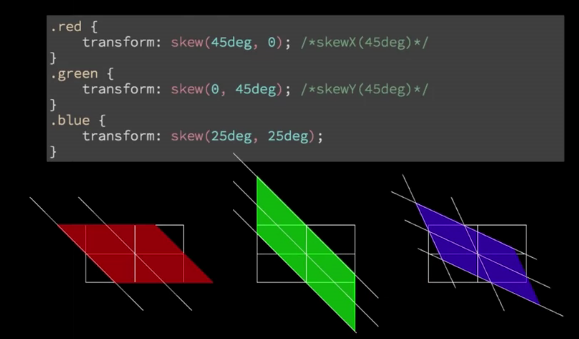
#### Наклон

Функция **skew(x,y)**

Где аргументы - это величина наклона по оси X и по оси Y.

Также есть функции skewX и skewY.

Значение задается в градусах

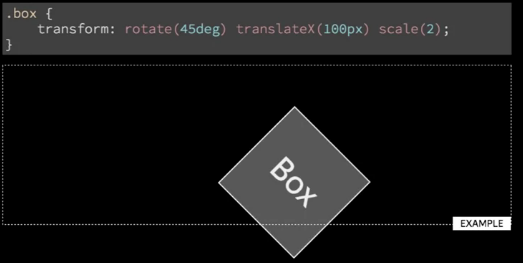


# Множественные преобразования

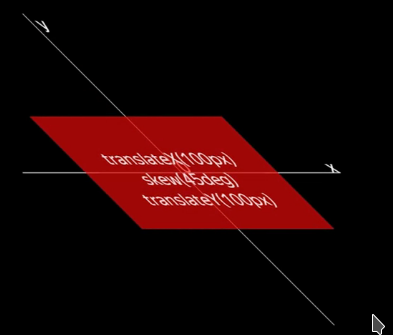
Множественные преобразования задаются функцией transform и через пробел указываются разные преобразования.



Все преобразования происходят последовательно, одно за другим



А тут нужно понимать, что после поворота, система координат тоже повернется, и последующее перемещение будет происходить по не совсем предполагаемой траектории.



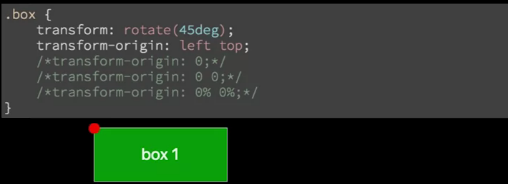
Нечто подобное происходить и при наклоне. Оси координат тоже сдвигаются.

#### Исходная точка

Поворот элемента происходить вокруг его центра, но положение этой точки можно изменить. Например:

**Transform-origin**

Кроме числовых(поценты, единицы длины и тп), у этого свойства есть значения **left, top, bottom, right.**

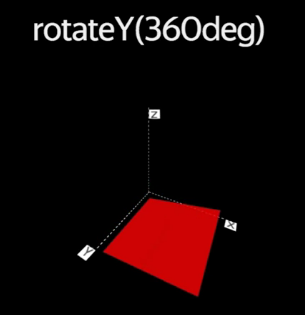


# Преобразования в трехмерном пространстве

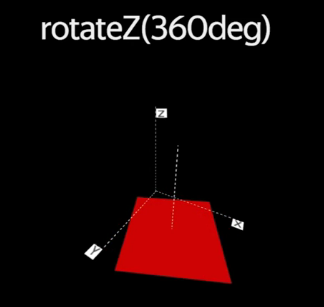
### Поворот

Функция **rotateX(360deg)** - поворот происходит не вокруг точки, а вокруг прямой - оси Х.

В свою очередь **rotateY** поворачивает объект вокруг оси Y



Ну и **rotateZ** - поворот вокруг оси Z

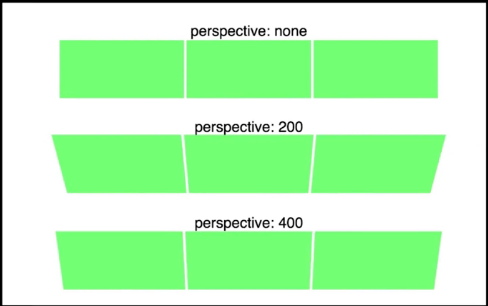


Для перемещения соответствующие функции называются - **translateX, translateY, translateZ.**

Функция **scale** для изменения размера осталась неизменной, потому что объекты в 3х мерном пространстве остаются плоскими, то есть не имеют толщины.

#### Перспектива

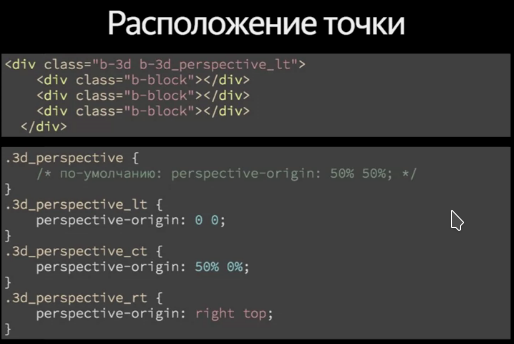
#### Для задания перспективы нужно использовать свойство **perspective** со значением, задающимся в пикселях



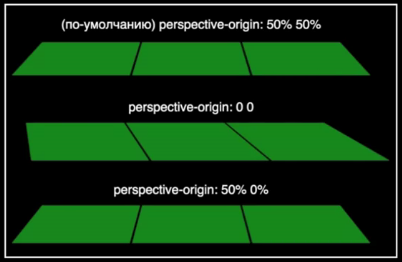
То есть значение - это глубина экрана. **None** - бесконечность.

#### Расположение точки.

Мы можем менять расположение точки схождения перспективы(????????).



По умолчанию эта точка находится посередине объекта



### Transform-style

Со значением **preserve-3d** задает трансформацию в трехмерном пространстве.



### Backface-visibility

Определяет видимость задней стороны объекта.



Тут оба куба на самом деле полупрозрачны



**Rotate3d(x, y, z, angle)** - первые три параметра это координаты вектор, последний - поворот

### Матрица преобразований

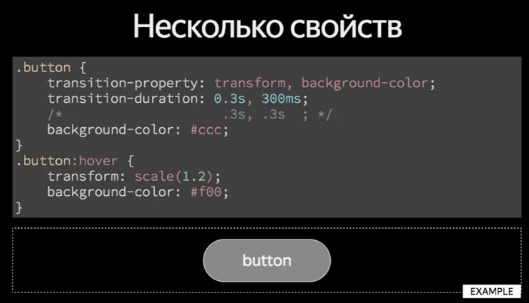
### 

Чаще всего используется для преобразований из жс

# Переходы

Переходы - анимация от одного набора свойств к другому. Для перехода необходимо:

1. Два набора свойств:
   1. Начальный
   2. Конечный
2. Свойство **transition** - описание свойств и характеристик анимации перехода.
3. **Инициатор** - действие, которое вызывает изменение от одного набора свойств к другому(:**hover, :target, :focus, :active и др**). Также инициатором может становится **js**, или добавление каких-то классов или идентификаторов.



Пример

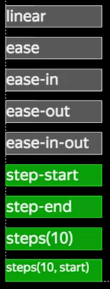
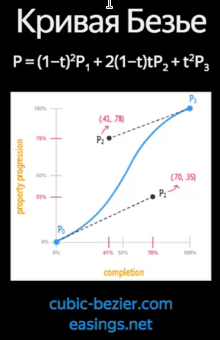
#### Задержка перехода

Свойство **transition-delay** и через запятую значения для каждого свойства, указанного в свойстве **transition-prorperty**. Может быть указано одно значение для всех свойств.



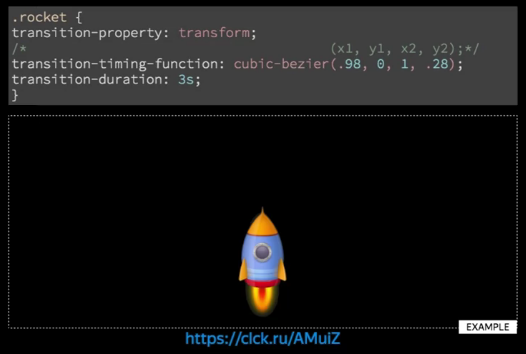
### Тип перехода

Свойство **transition-timing-function** с значениями:

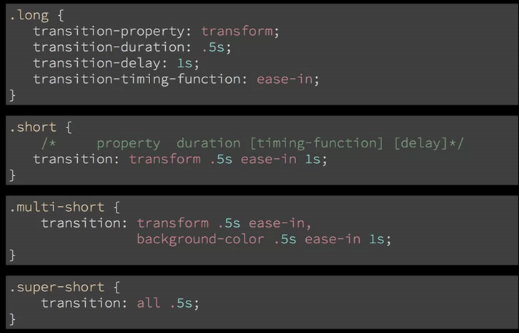
 

Можно указывать также свои функции

Пример:



### Короткая запись для этих свойств выглядит так:



Можно указывать несколько наборов свойств через запятую.

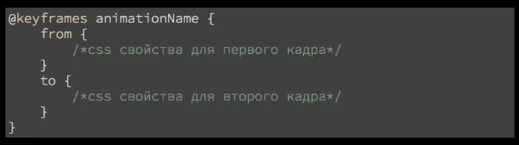
# Анимации

Свойство **animation** - позволяет анимировать переходы между ключевыми кадрами.

Для создания анимации необходимо:

1. Определить ключевые кадры - содержат свойства, которые применяются в определенный момент времени при анимации.
2. Применение анимации к элементу.

Нужно задать at правило @keyframes и задать ему какое-то имя:

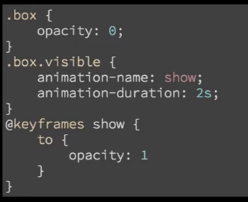


Прмер:



Тут свойство **animation-name** задает имя првила, а **animation-duration -** время анимации. **Animation-delay** - установка задержки перед анимацией.

Еще пример:



Стоит учесть что ДО и ПОСЛЕ анимации элемент приходит к состоянию, которое было у него по умолчанию до анимации. То есть в первом примере элемент будет изначально видимый, потом исчезнет и плавно появится. А во втором примере элемент исчезнет после анимации.

Одну и ту же анимации можно применять к разным объектам.

### Использование промежуточных значений



То есть будет промежуточное значение посередине анимации

Также можно использовать в записи ТОЛЬКО проценты



Также можно указывать через запятую те состояния, к которым нужно применить одинаковые свойства

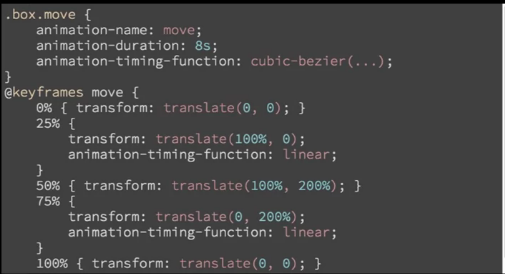


Анимировать можно одновременно несколько свойств

### Тип анимации

Как и в переходах, в анимациях можно задавать тип, то есть функцию или ключевое слово.

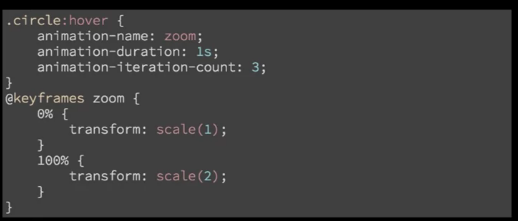
За это отвечает свойство **animation-timing-function**:



При этом можно указывать функцию как у объекта, так и непосредственно внутри ключевых кадров. **Указанная внутри ключевого кадра функция применяется только для одного временного отрезка.**

#### Повторение анимации

Свойство **animtion-iteration-count** у объекта:



**Infinite** - значение для бесконечного повторения

Свойство **animation-direction** у объекта:

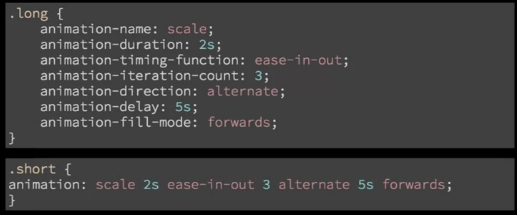
1. **Alternate** - при повторении анимации, при достижение конца одной анимации, состояние объекта возвращается обратно тем же путем, однако счетчиком это возвращение считается за повторение.

#### Скачки после завешения

Свойство **animation-fill-mode** с значением **forwards:**

Тогда после завершения анимации объекта он застывает в конечном состоянии.

#### Краткая запись:



Можно указывать несколько анимаций через запятую

#### Управление анимацией

Можно ставить на паузу

