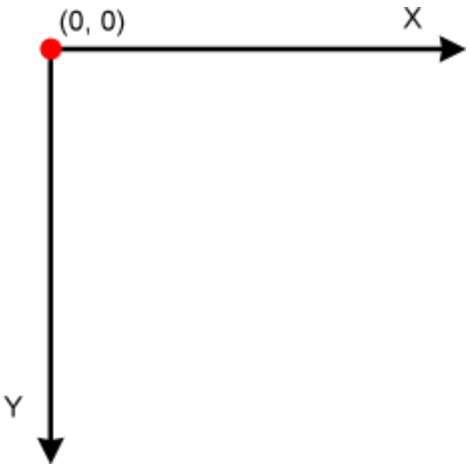
# Перемещение по горизонтали

В этой части мы рассмотрим возможности двухмерных трансформаций в CSS.

То, что раньше можно было сделать в окне браузера только с помощью JavaScript — плавное перемещение и масштабирование блоков, повороты и наклоны — теперь с лёгкостью реализуется на чистом CSS.

Для начала давайте разберёмся с системой координат, в которой перемещается объект:




её ключевой особенностью является то, что ось Y направлена вниз, а не вверх, так как веб-страница начинается с левого верхнего угла и идет вниз, а в CSS используется обратная система координат.

Первое перемещение, которое мы осуществим — горизонтальное. Мы будем двигать объекты влево и вправо.

Для этой и других трансформаций используется следующий синтаксис:

transform: функция трансформации(значение трансформации)

Горизонтальное перемещение осуществляется функцией translateX. Значение функции трансформации численное, а возможные единицы измерения — px, %, em или in. Например, такая функция переместит объект на 100 пикселей вправо по оси X:

transform: translateX(100px)

# Перемещение по вертикали

Для этого воспользуемся похожей на «горизонтальную» функцией `translateY`:

transform: translateY(-100px)

Такая функция переместит объект на 100 пикселей вертикально вверх. Заметьте, что для движения вверх используется отрицательное значение трансформации.

К объекту может быть одновременно применено несколько функций трансформации. При этом функции просто перечисляются через пробел после имени свойства transform, например:

transform: translateY(-100px) translateX(100px)

Такая трансформация переместит объект на 100 пикселей вправо и вверх по оси координат.

```
transform: translate(перемещение по оси X [, перемещение по оси Y])
```

Квадратные скобки указывают на то, что значение перемещения по оси Y является необязательным аргументом. В коде квадратные скобки писать не нужно, два значения просто перечисляются через запятую, причём у них могут быть разные единицы измерения. Если не указать значение перемещения по оси Y, а написать translate(перемещение по оси X), то значение перемещения по Y будет считаться равным 0 и функция будет работать аналогично translateX.

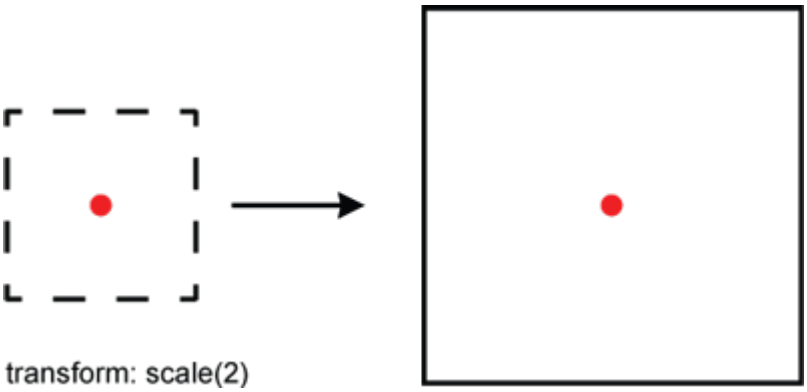
```
// Эти функции работают одинаково:

transform: translate(100px);

transform: translate(100px, 0);

transform: translateX(100px);
```

## Увеличение, уменьшение

Ещё одной возможностью CSS-трансформаций является масштабирование блоков. С помощью функции `scale` можно увеличивать или уменьшать элементы.

Значением этой функции может быть любое число. При этом значение `1` считается точкой отсчёта, когда блок имеет оригинальные размеры. Рассмотрим примеры:

- `scale(0.5)` уменьшит объект в 2 раза;
- `scale(2)` увеличит объект в 2 раза;
- `scale(0)` полностью «схлопнет» объект, и его не будет видно;
- `scale(1)` оставит объект без изменений.



transform: scale(2)

Вообще функция `scale`, как и `translate`, принимает 2 аргумента:

```
scale(масштаб-по-X [, масштаб-по-Y])
```

Если необязательный аргумент `масштаб-по-Y` не задан, то считается, что он такой же, как `масштаб-по-X`:

```
transform: scale(2) то же самое, что transform: scale(2, 2)
```

Также в случае, когда необходимо масштабировать объект только по одной оси независимо от другой, существуют функции `scaleX(масштаб-по-X)` и `scaleY(масштаб-по-Y)`.

## Наклоны

Ещё одна возможность CSS-трансформаций — наклон объекта по осям X и Y под заданным углом. Наклоны создаются с помощью функций `skewX` и `skewY`.

Угол наклона задаётся в градусах — `deg`. Например:

```
transform: skewX(45deg)

transform: skewY(30deg)
```

Для оси X положительное значение угла наклоняет объект влево, а отрицательное — вправо. Для оси Y — вниз и вверх соответственно.

Заметим, что наряду со skewX и skewY существует обобщающая функция skew, которая принимает два аргумента: skew(наклон-по-X [, наклон-по-Y]). Значение наклона по оси Y является необязательным аргументом и по умолчанию равно 0. Но при этом поведение функции skew при трансформации отличается от одновременного применения skewX и skewY. Исторически так сложилось, что реализация skew в браузерах поддерживается сейчас только для совместимости прежнего контента, и даже исключалась из рабочего черновика спецификации CSS. В общем, вместо skew лучше пользоваться skewX и skewY.

## Особенности transform-origin

Это свойство используется совместно со свойством `transform` и задаёт точку отсчёта системы координат, в рамках которой будет работать трансформация.

Синтаксис `transform-origin` для двухмерных трансформаций следующий:

```
transform-origin: точка-отсчёта-по-X [точка-отсчёта-по-Y]
```

Значения свойства задаётся в единицах измерения ширины в браузере (`px`, `em`...), в `%`, а также ключевыми словами `left`, `right`, `top`, `bottom` и `center`.

По умолчанию, значение `transform-origin` равно `50% 50%`, то есть начало системы координат находится в центре объекта. Если не указывать значение `точка-отсчёта-по-Y`, то оно считается равным `50%`.

А теперь попробуем сделать трансформацию плавной. Для этого в CSS предусмотрено свойство transition.

Если в двух словах, transition позволяет изменить значение какого-либо свойства плавно.

## Центровка с помощью transform: translate

Для центровки одного блока внутри другого блока обычно используют классический трюк с позиционированием, относительными координатами и отрицательными маргинами.

Этот приём хорошо работает, когда центруемый блок имеет фиксированные размеры, но если его размеры могут изменяться, то возникают проблемы.

С помощью трансформаций можно решить эту проблему и центровать блоки с переменными размерами. Делается это с помощью translate.

## Нестандартные тени

Иногда бывает нужно реализовать тени, отличающиеся от стандартных `box-shadow`, без применения картинок.

Например, тени, наклонённые в разные стороны.

Сделать это с помощью `box-shadow` невозможно. Поэтому нужно использовать более сложный приём:

- с помощью псевдоэлементов `before` и `after` создаём два блока с обычными тенями;
- наклоняем эти блоки с помощью `rotate`;
- задаём им отрицательный `z-index`.

Это переместит блоки с тенями под родительский контейнер так, что наружу будут выглядывать только кусочки теней.

## Эффекты при наведении: кнопки

Довольно много интересных эффектов можно добиться с помощью трансформаций при реализации разных элементов интерфейса, например, кнопок. Им можно добавить немного динамики, используя transform совместно с transition для создания простейшей плавной анимации.

В любой ситуации, когда стоит задача плавно менять позицию или размеры блока, а также каким-либо способом менять его внешний вид, встаёт выбор: пользоваться связкой transition + transform или же задействовать javascript и динамически менять ширину/высоту, координаты объектов в скрипте.

Зачастую выбор падает на JS-реализацию с помощью библиотеки jQuery, но в большинстве случаев «чистая» CSS-реализация простых визуальных эффектов намного производительнее аналогичных jQuery-функций. Кроме того, в ряде случаев определённые трансформации в CSS могут для повышения производительности задействовать не только центральный процессор компьютера или мобильного девайса, но и ресурсы графического адаптера, что позволяет разгрузить процессор и избавиться от «тормознутости» эффектов.