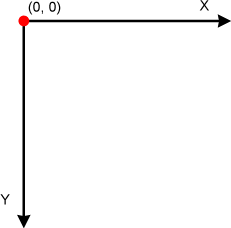
**Перемещение по горизонтали**

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](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D1%8F%D0%BC%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82).

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

Теперь давайте попробуем перемещать объекты 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.

Как вы уже увидели в предыдущем задании, к объекту может быть одновременно применено несколько функций трансформации. При этом функции просто перечисляются через пробел после имени свойства transform, например:

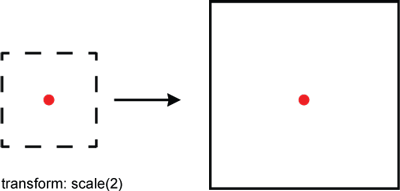
transform: translateY(-100px) translateX(100px)

**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.



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-Х) и scaleY(масштаб-по-Y).

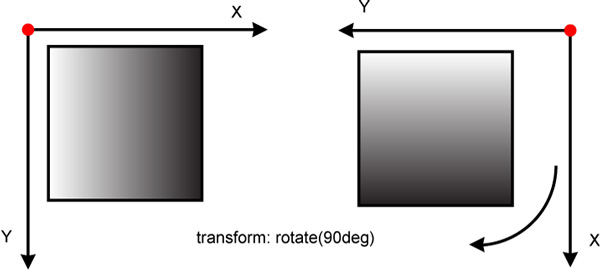
Для этого нужно собраться и быстро вспомнить всё, а также освоить пару новых приёмов.



В этом задании нам понадобится трансформация c функцией поворота transform: rotate. Её синтаксис: transform: rotate(угол наклона).

Положительное значение угла повернёт объект по часовой стрелке, а отрицательное — против. Единица измерения поворота — градусы deg, например: transform: rotate(180deg) повернёт объект на 180° по часовой стрелке, то есть перевернёт его.

При совместном применении трансформации поворота и перемещения есть небольшая тонкость. При повороте вместе с объектом на заданный угол поворачивается и вся его система координат:



То есть, чтобы повёрнутый на 90° блок двигать по горизонтали, нужно применять translateY, а по вертикали — translateX.

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.

В примере ниже первый блок наклонён на 25° влево, а второй — на 25° вниз:

transform: skewX(25deg);transform: skewY(25deg);

Заметим, что наряду со skewX и skewY существует обобщающая функция skew, которая принимает два аргумента: skew(наклон-по-Х [, 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.

ещё одну интересную особенность transform: scale. А именно то, как ведёт себя объект при задании отрицательных числовых значений: при scaleX(-1), например, оригинальный объект зеркально отражается по горизонтали, а при scale(-1, -1) — отражается и по оси X, и Y. Большее, чем -1 значение ещё шире растянет отзеркалированный объект.

Ниже представлены примеры подобных трансформаций:

transform: scaleX(1);  
transform: scaleX(1);transform: scaleX(-1);  
transform: scaleX(-1);transform: scale(-1);  
transform: scale(-1);transform: scaleX(-1.5);  
transform: scaleX(-1.5);

# Особенности transform-origin — часть 1

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%.

Закрыть

* Теория

**Особенности transform-origin — часть 2**

А теперь попробуем сделать трансформацию плавной. Для этого в CSS предусмотрено свойство transition. Не будем пока вдаваться в подробности, как именно работает и применяется transition — это будет подробно рассмотрено в ближайшем тренажёре.

Если в двух словах, transition позволяет изменить значение какого-либо свойства плавно. В нашем случае плавно будет меняться свойство transform с функцией scale, а переход между двумя значениями будет длиться полсекунды.

transition: transform 0.5s ease;

Центрирование объекта

.picture {

position: relative;

top: 50%;

left: 50%;

width: 50%;

height: 50%;

transform: translateX(-50%) translateY(-50%);

}

**Круговое меню, финал**

Вот и вся техника для создания круглого меню:

1. Переносим ось вращения пунктов в угол с помощью transform-origin и совмещаем этот угол с центром меню.
2. Наклоняем и поворачиваем пункты меню, используя формулы:
3. [угол сектора] = 360° / [количество пунктов меню]
4. [поворот пункта n] = (n - 1) \* [угол сектора]

[угол наклона] = 90° - [угол сектора]

1. «Разнаклоняем» и разворачиваем содержимое пунктов меню:
2. [угол наклона содержимого] = -1 \* [угол наклона пункта]

[угол разворота] = -1 \* (90° – ([угол сектора] / 2))

1. Смещаем содержимое пунктов к центру круга.
2. Закругляем контейнер меню с помощью border-radius и обрезаем всё лишнее с помощью overflow: hidden.

А после этих шагов можно применять дополнительное оформление. Например, задавать разные фоны пунктов, рамку для контейнера меню и так далее.

Давайте напоследок добавим третий пункт меню, чтобы у нас получился полукруг. Угол поворота третьего пункта будет равен 120°: