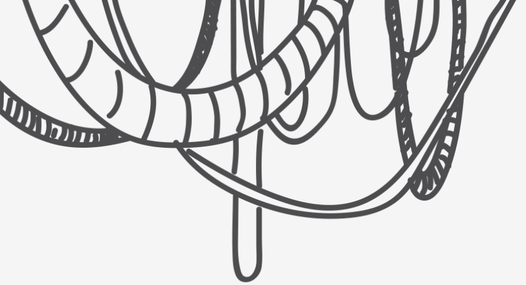
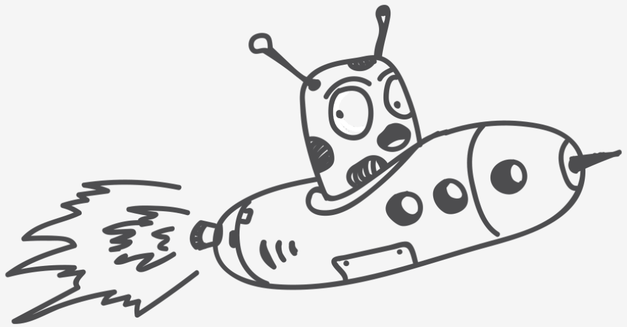
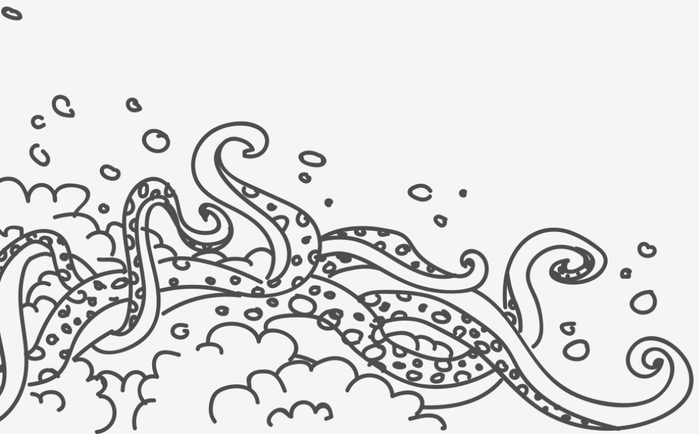






Transformacje i efekty przejścia

Transformacje

Czym to jest?

Transformacje pozwalają na przekształcanie elementów poprzez skalowanie, pochylenie, przemieszczanie i obrót.

Transformacje mogą być wykonane w dwóch **(2D)** lub trzech wymiarach **(3D)**.

Właściwość ta posiada dobre wsparcie we wszystkich nowoczesnych przeglądarkach (http://caniuse.com/#search=transforms).

Transformacje działają na element poddawany przekształceniu jak również na elementy znajdujące się wewnątrz modyfikowanego rodzica.

Obracanie

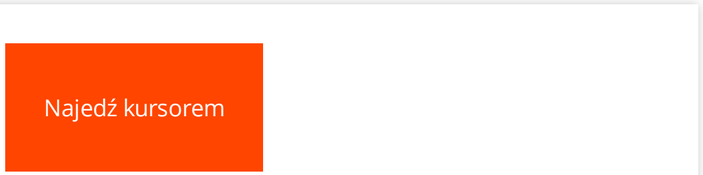
Właściwość `rotate` pozwala na obracanie elementu względem własnej osi.

Może przyjmować wartości dodatnie i ujemne wyrażone w stopniach (`deg`), radianach (`rad`) i obrotach (`turn`).

CSS

```
.test {
  width: 200px;
  height: 100px;
  background: orangered;
  margin: 10px auto;
  transition: 0.5s;
}

.test:hover {
  transform:rotate(45deg);
}
```

Przykład



Obracanie

CSS

```
.test {
  width: 200px;
  height: 100px;
  background: orangered;
  margin: 10px auto;
  transition: 0.5s;
  display: flex;
  justify-content: center;
  align-items: center;
}

.test .child {
  width: 50%;
  height: 50%;
  transition: 0.5s;
  background: dodgerblue;
}

.test:hover {
  transform:rotate(90deg);
}

.test:hover .child {
  transform: rotate(-180deg);
}
```

Pamiętajmy, że jeżeli obrócimy dany element, wraz z nim obrócimy jego dzieci. Równocześnie możemy też obracać jego dzieci indywidualnie.

Przykład

Przemieszczanie

Właściwość `translate` pozwala na przemieszczenie elementu w przestrzeni 2D.

Przemieszczenie może się odbywać względem osi `X (translateX)`, `Y (translateY)` lub obu `(translate)`.

CSS

```
.test {
  width: 200px;
  height: 100px;
  background: orangered;
  margin: 10px auto;
  transition: 0.5s;
}

.test:hover {
  transform: translate(50%, 50%);
}
```

Przykład

Przemieszczanie

W przypadku tej właściwości możemy używać dowolnej jednostki.

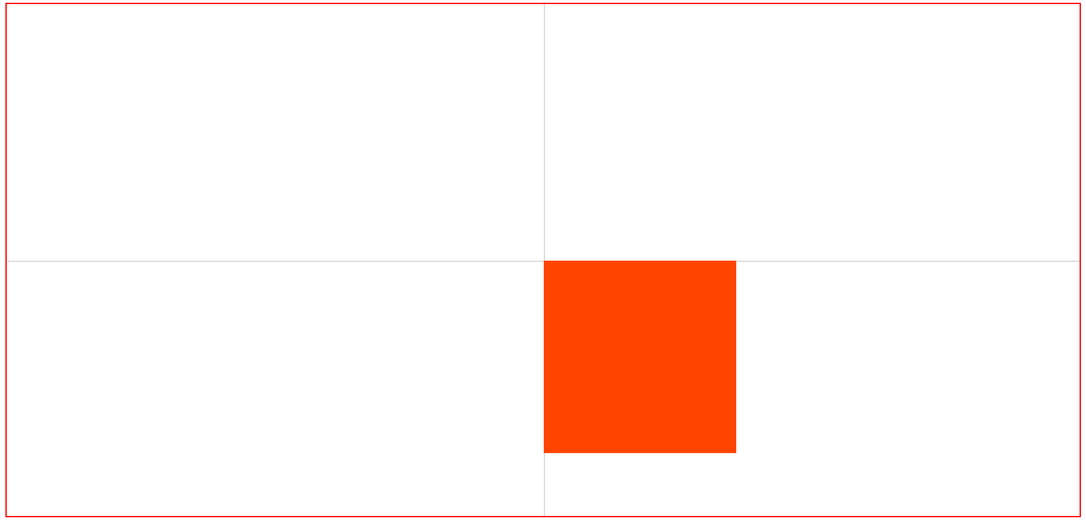
Istnieje tutaj jednak mała różnica w przypadku jednostki %.

Jeżeli odsuwamy element za pomocą właściwości `margin-top`, `margin-left` lub `left/top/bottom/right` (wraz z `position`) i przesunięcie to podajemy w %, odnosi się ono do rozmiaru rodzica. Dla przykładu `margin-top`: 50% oznacza, że przesuniemy dany element 50% wysokości rodzica.

W przypadku transition % odnosi się do danego elementu, czyli transform: `translate(50%)` przesunie element o 50% szerokości danego elementu.

Przykład 1

```
position: absolute;
top: 50%;
left: 50%;
```



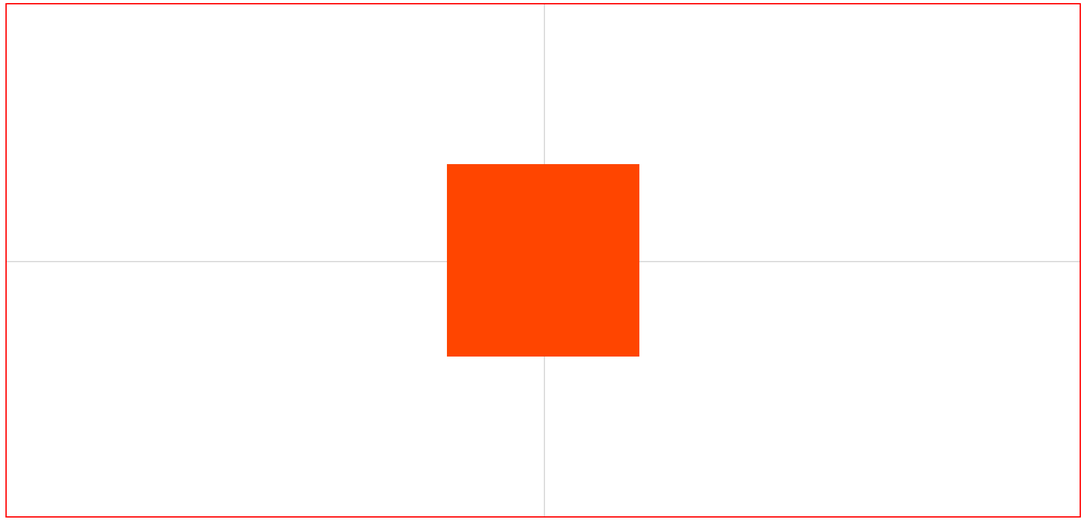
Przemieszczanie

Przykład 2

```
position: absolute;
top: 50%;
left: 50%;
margin-left: -50%;
margin-top: -50%;
```

Przykład 3

```
position: absolute;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
```



Skalowanie - scale

Dzięki właściwości `scale` możemy zmienić rozmiar elementu (przeskalować go) o dowolną wielkość wyrażoną mnożnikiem.

Wartości poniżej 1 zmniejszają nam element. Wartości wyższe niż 1 skalują dany element w górę.

CSS

```
.test {
  width: 200px;
  height: 100px;
  background: orangered;
  margin: 10px auto;
  transition: 0.5s;
}

.test:hover {
    transform: scale(1.5);
}
```

Przykład

Najedź kursorem

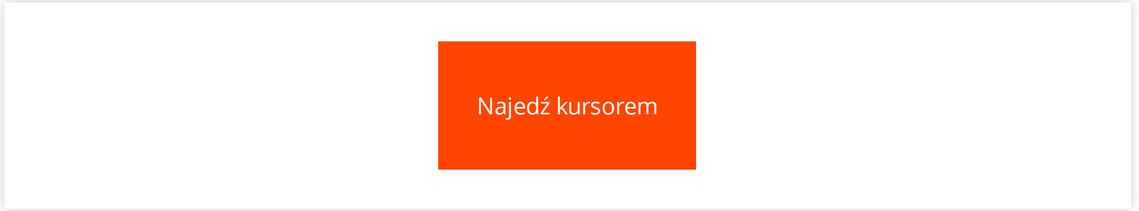
Pochylenie

Właściwość `skew` pozwala na pochylanie elementu względem osi X lub Y (lub obu jednocześnie).

Zakres pochylenia elementu wyrażamy w stopniach (deg).

Przykład

```
.test:hover {
  transform: skew(-35deg);
}
```



Łączenie transformacji

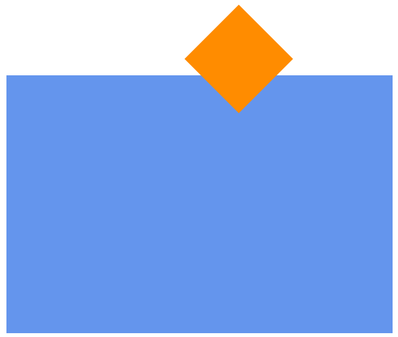
W ramach jednej reguły możemy wykonać jednocześnie **kilka rodzajów transformacji** dla danego elementu np. jednocześnie przeskalować, pochylić i obrócić element.

```
.test {
  transform: scale(1.6) skew(-20deg) rotate(10deg);
}
```

W przypadku łączenia transformacji ważna jest kolejność!

W przykładzie obok, autor pomylił kolejność. Na początku obrócił obiekt o 45deg, a następnie go przemieścił (czyli przemieścił już po obróconych kierunkach).

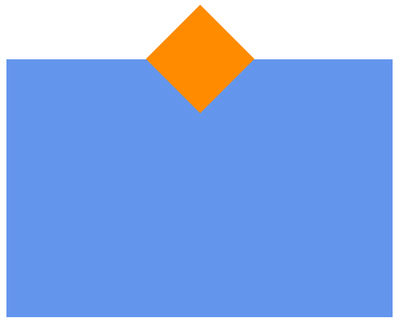
```
.test {
  width: 300px;
  height: 200px;
  background: cornflowerblue;
  position: relative;
}
.test:after {
  content: '';
  width: 60px;
  height: 60px;
  background: darkorange;
  position: absolute;
  left: 50%;
  top: 0;
  transform: rotate(45deg) translate(-50%, -50%);
}
```



Łączenie transformacji

W kolejnym przypadku autor na szczęście już się poprawił i jak widzimy czerwony obrócony kwadrat jest już idealnie na środku swojego niebieskiego rodzica.

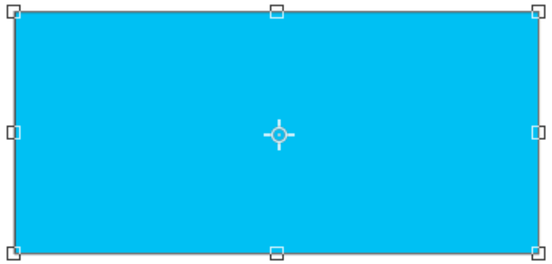
```
.test {
  width: 300px;
  height: 200px;
  background: cornflowerblue;
  position: relative;
}
.test:after {
  content: '';
  width: 60px;
  height: 60px;
  background: darkorange;
  position: absolute;
  left: 50%;
  top: 0;
  transform: translate(-50%, -50%) rotate(45deg);
}
```



Transform-origin

Właściwość `transform-origin` pozwala nam na ustalenie punkt, względem których ma odbyć się transformacja.

Jeżeli używaliśmy kiedyś transformacji w programie photoshop lub podobnym, właściwość ta nie będzie nam obca:



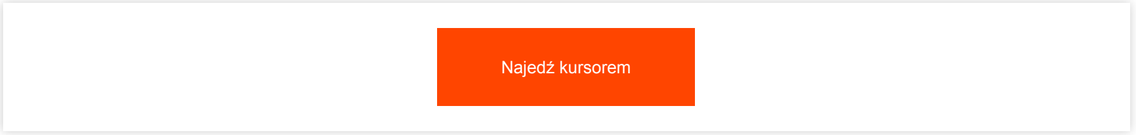
Domyślnie transformacje przeprowadzane są względem środka obiektu (50%, 50%). Punkt ten możemy zmieniać za pomocą dowolnej jednostki. Najczęściej używaną będą procenty.

Efekty przejścia - transition

Właściwość `transition` pozwala nam wskazać właściwości, które jeżeli w przyszłości się zmienią, **nastąpi to płynnie w określonym przez nas czasie**.

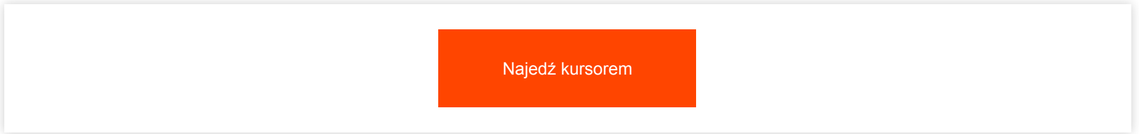
Bez transition

```
button {
  background: orangered;
  width: 200px;
  height: 60px;
  color: #fff;
  border: 0;
}

button:hover {
  background: dodgerblue;
}
```



Z transition

```
button {
  background: orangered;
  width: 200px;
  height: 60px;
  color: #fff;
  border: 0;
  transition: 0.5s;
}

button:hover {
  background: dodgerblue;
}
```



transition-duration

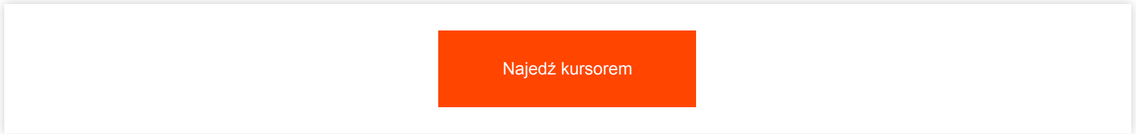
Czyli czas trwania przejścia z jednego stanu na drugi. Domyślnie wynosi 0s.

Czas może być podawany w sekundach (s) lub milisekundach (ms).

CSS

```
button {
  transition-duration: 2s;
}
button:hover {
  border-radius: 50px;
  background: dodgerblue;
}
```

Przykład



transition-delay

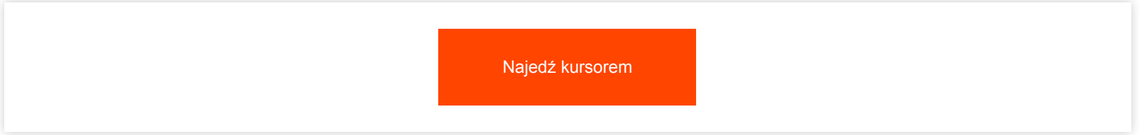
Czyli czas opóźnienia. Domyślnie właściwość ta wynosi 0s.

Czas może być podawany w sekundach (s) lub milisekundach (ms).

CSS

```
button {
  transition-duration: 2s;
  transition-delay: 0.5s;
}
button:hover {
  border-radius: 50px;
  background: dodgerblue;
}
```

Przykład



transition-property

Domyślnie po użyciu `transition` na elemencie wszystkie właściwości, które mogą być “animowane” będą podlegać płynnej zmianie.

Czasami jednak nie będzie to dobrym rozwiązaniem. W takich przypadkach możemy wskazać o jaką właściwość nam chodzi.

Możemy też tutaj użyć słowa kluczowego all (domyślnie), które oznacza wszystkie właściwości a dodatkowo zapisać to w jednej właściwości razem z czasem trwania.

```
button {
  transition: 1s all;
}
```

CSS

```
button {
  transition-duration: 5s;
  transition-property: border-radius;
}

button:hover {
  border-radius: 50px;
  background: dodgerblue;
}
```

Przykład