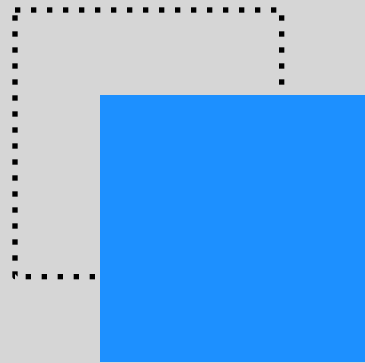


CSS challenge #2

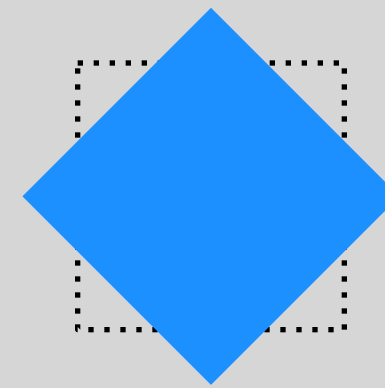
TRANSFORM & TRANSITION INTRO

Individual transform properties

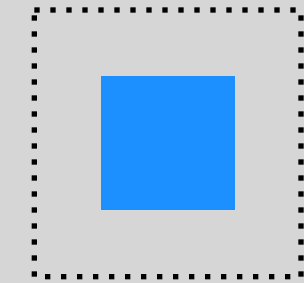
`translate: 2em 2em;`



`rotate: 45deg;`

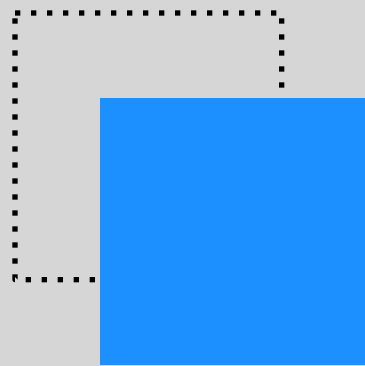


`scale: .5;`



Individual transform properties

translate: 2em 2em;



min is
omhoog

X-as Y-as

translate: 2em -2em;

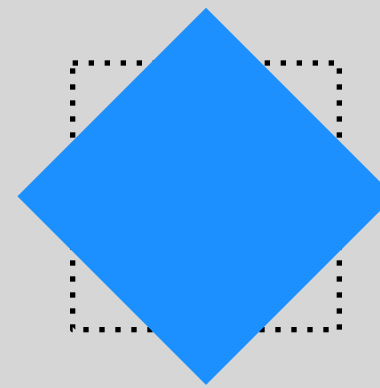
X-as

translate: 2em;

X-as Y-as Z-as

translate: 2em 2em 2em;

rotate: 45deg;



Z-as

rotate: 90deg;

X-as

rotate: x 120deg;

Y-as

rotate: y 1turn;

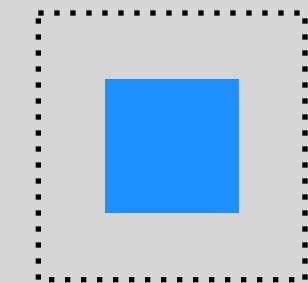
Z-as

rotate: z -90deg;

tegen de
klok in

1 draai

scale: .5;



2x zo
groot

X&Y-as

scale: 2;

helemaal
weg

scale: 0;

X-as Y-as

scale: .5 2;

scale: 50% 200%;

percentages
kunnen ook

Transition

```
transition-duration: 1s;  
transition-delay: 1s;  
transition-timing-function: ease-in;
```

Kan samen ook met de shorthand:

```
transition: 1s;  
transition: 1s 1s ease-in;  
delay
```

Alleen van toepassing voor color:

```
transition-property: color;
```

Als onderdeel van de shorthand:

```
transition: color 1s 1s ease-in;
```

Verschillende transitities:

```
transition:  
  color 1s .5s ease-in,  
  border-radius 2s 1s ease-in;
```

custom easing

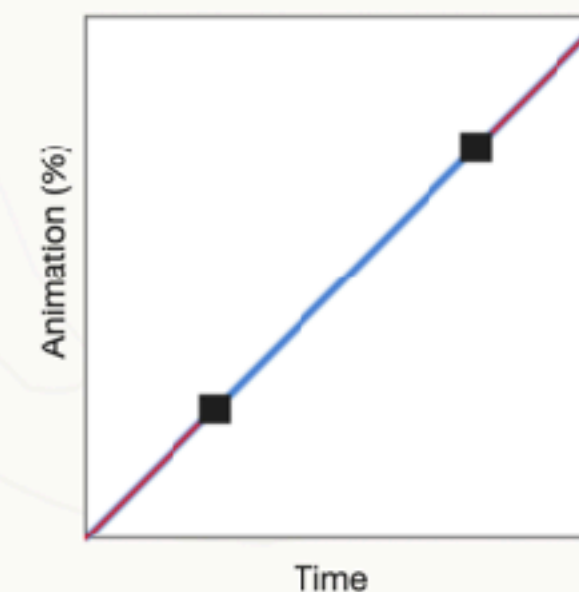


CSS EASING ANIMATION TOOL

1. Choose an easing type and test it out with a few effects.
2. If you don't quite like the easing, grab a handle and fix it.
3. When you're happy, snag your code and off you go.

Now that we can use CSS transitions in all the modern browsers, let's make them pretty. I love the classic Penner equations with Flash and jQuery, so I included most of those. If you're anything like me*, you probably thought this about the default easing options: "ease-in, ease-out...yawn." The mysterious cubic-bezier has a lot of potential, but was cumbersome to use. Until now. Also, touch-device friendly!

*If you are anything like me, we should be friends [@matthewlein](#)



Easing:

Duration:

Effect:



Code snippets, short and long-hand:

```
transition: all 500ms cubic-bezier(0.250, 0.250, 0.750, 0.750); /* linear */
```

```
transition-timing-function: cubic-bezier(0.250, 0.250, 0.750, 0.750); /* linear */
```

If this saves you time, or blows your mind, consider making a to keep these projects alive.

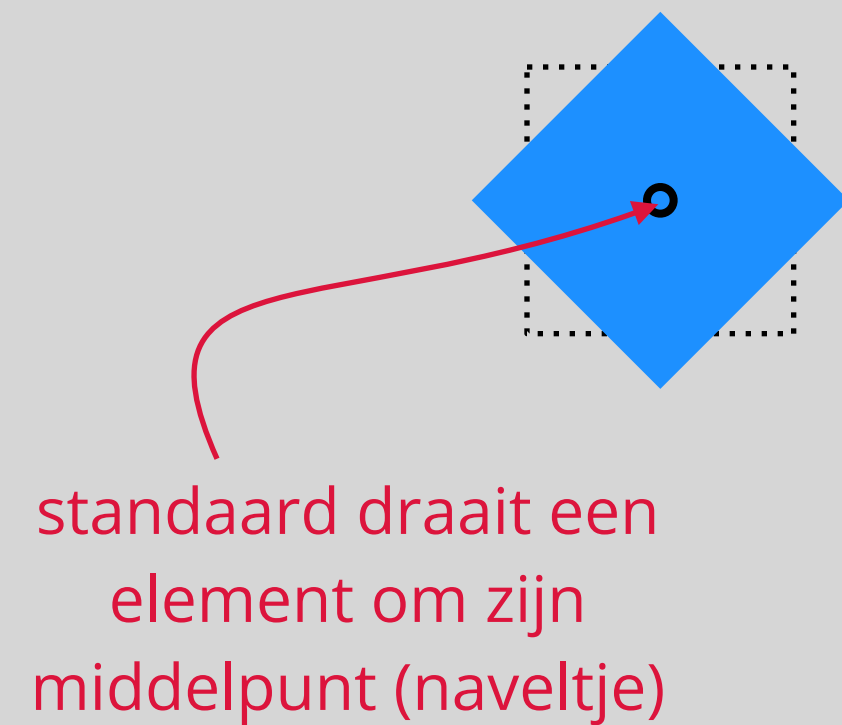
Resources

- Very nice overview of CSS Transition

<https://matthewlein.com/tools/ceaser>

Transform-origin

rotate: 45deg;



rotate: 135deg;



rotate: 135deg;



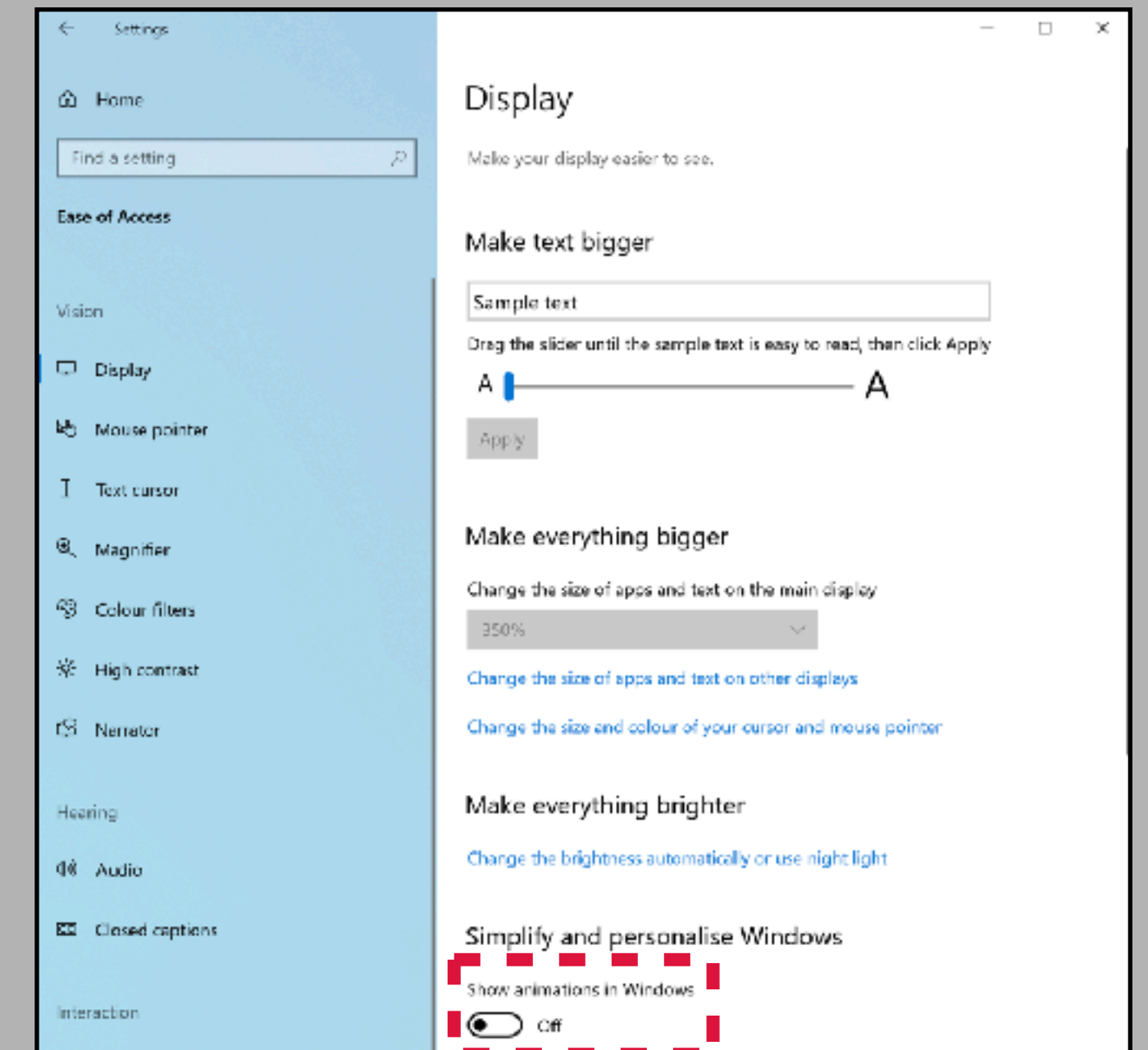
Prefers reduced motion

```
@media (prefers-reduced-motion:no-preference) {  
  div {  
    transition: 1s;  
  }  
}
```

alleen een transition als de
gebruiker heeft aangegeven
dat dat geen probleem is

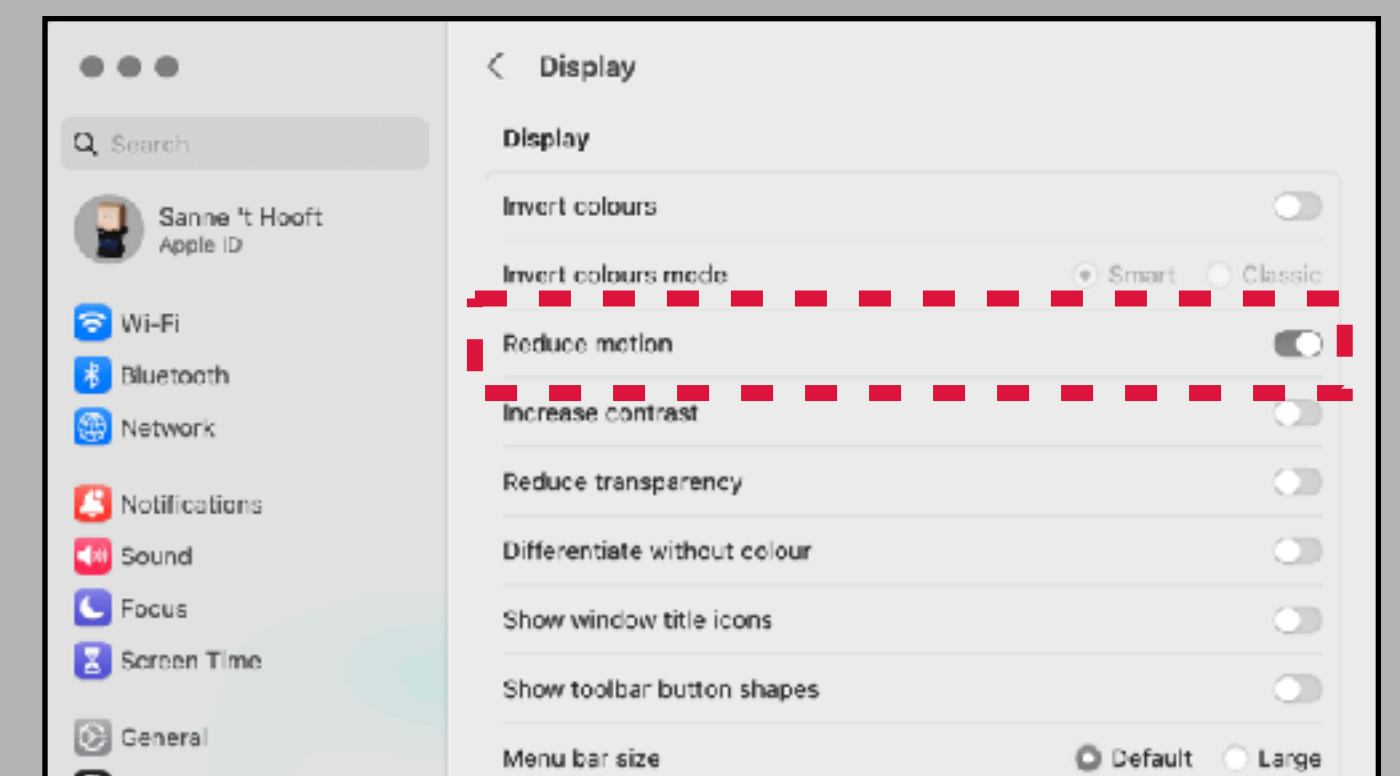
Show animations - Windows

Settings → Ease of Access → Display



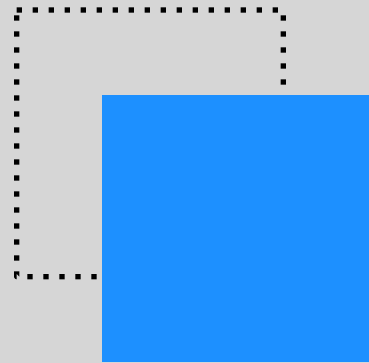
Reduce motion

System preferences → Accessibility → Display

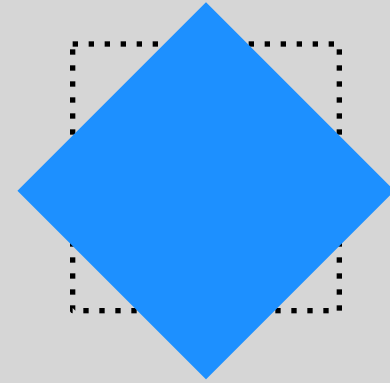


Transform functions

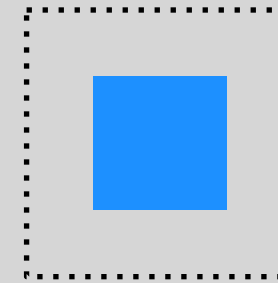
`transform: translate(2em, 2em);`



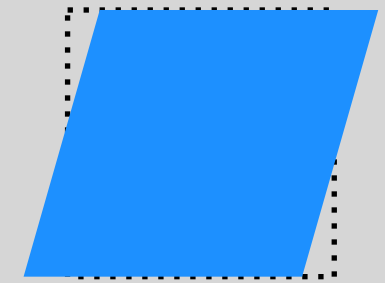
`transform: rotate(45deg);`



`transform: scale(.5);`

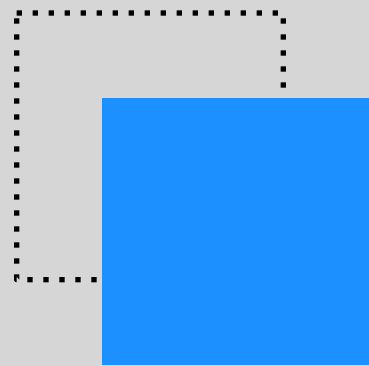


`transform: skew(10deg, 0deg);`



Transform functions

`transform: translate(2em, 2em);`



percentage
van eigen
afmeting

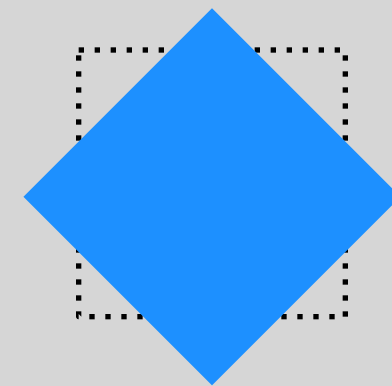
X-as Y-as
`translate(2em, -2em);`

X-as
`translate(2em);`

`translate(50%, 3rem);`

`translateY(100%);`
`translateX(100%);`
`translateZ(100%);`

`transform: rotate(45deg);`

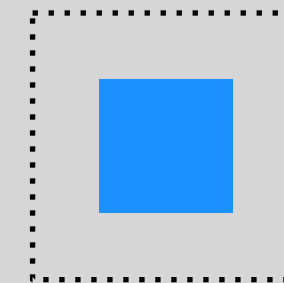


Z-as
`rotate(45deg);`

`rotateX(.5turn);`
`rotateY(0);`
`rotateZ(45deg);`

`rotate3D(1,1,1,45deg);`

`transform: scale(.5);`



X&Y-as

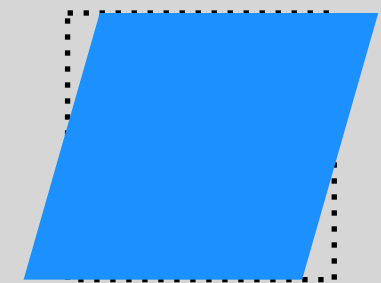
`scale(1);`

X-as Y-as

`scale(.5, 2);`

`scale(50%, 200%);`

`transform: skew(10deg, 0deg);`



X-as

Y-as

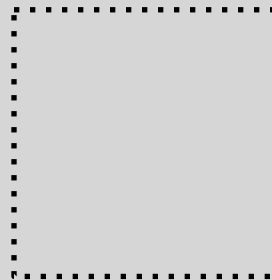
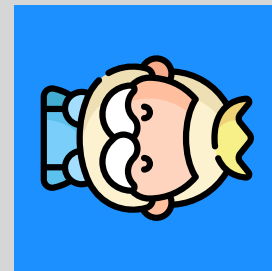
`skew(10deg, 20deg);`

X-as

`skew(15deg);`

`skewX(.25turn);`
`skewY(-10deg);`

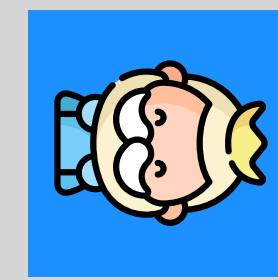
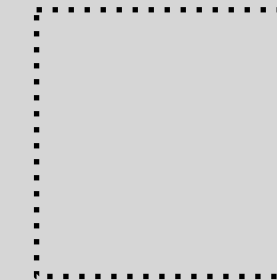
Combi (de volgorde doet ertoe)



transform:
translateY(-125%)
rotate(90deg);

eerst omhoog

dan draaien



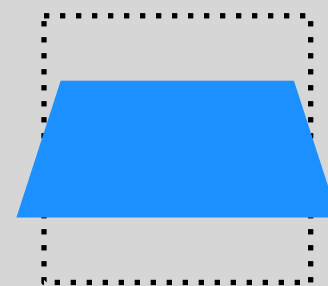
transform:
rotate(90deg)
translateY(-125%);

eerst draaien
(het assenstelsel
draait mee)

dan 'omhoog' (in het
gedraaide assenstelsel)

3D

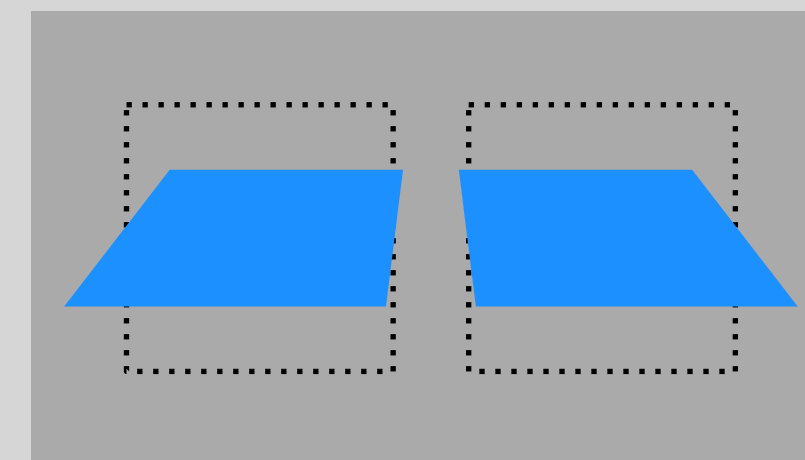
Eigen perspectief



```
div {  
  transform:  
    perspective(15em)  
    rotateX(45deg);  
}
```

de div heeft zijn eigen
verdwijnpunt

Gedeeld perspectief

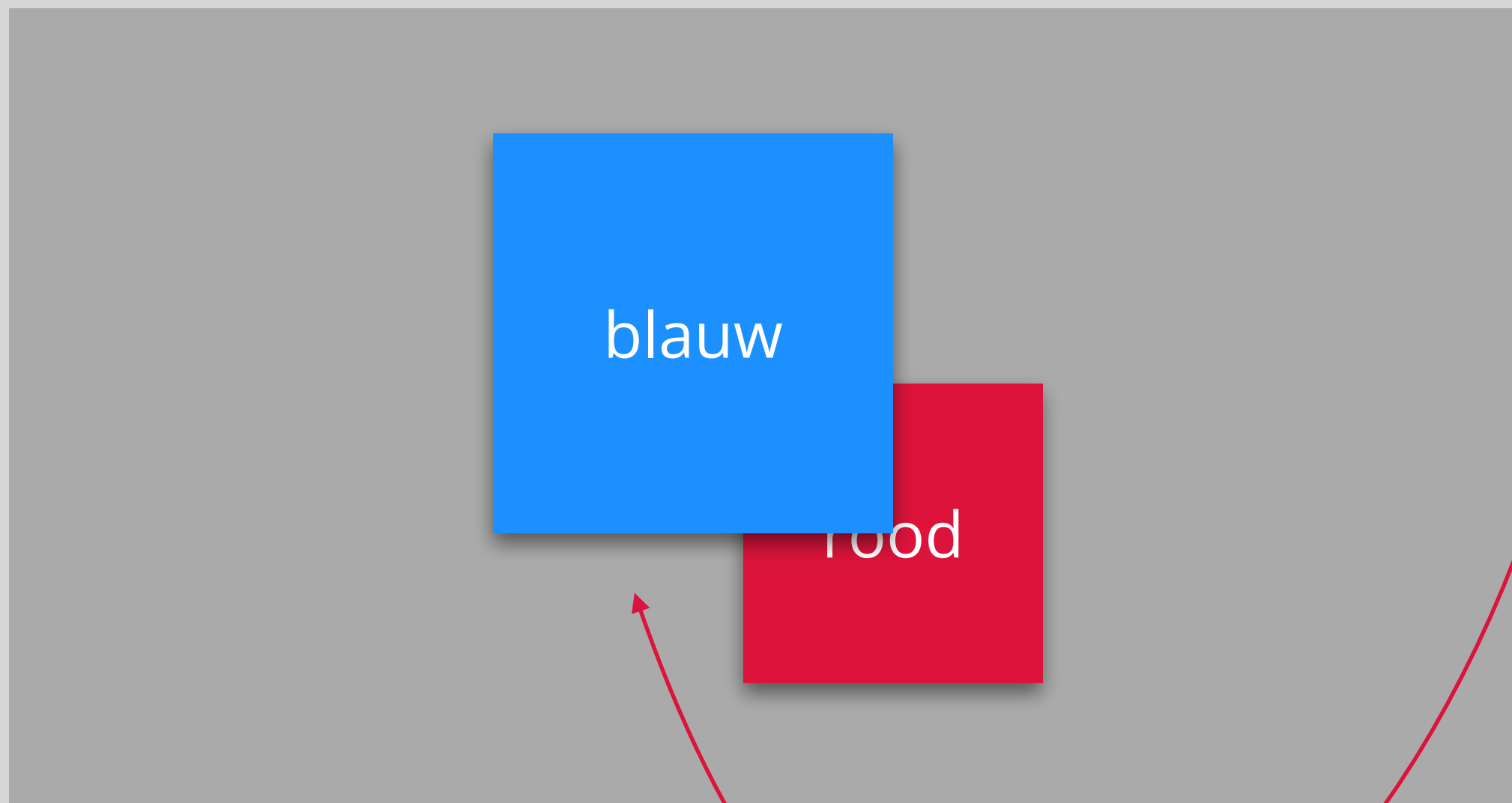


```
section {  
  perspective:15em;  
}  
  
div {  
  transform:  
    rotateX(45deg);  
}
```

de divs delen hetzelfde
verdwijnpunt van de
section

preserve-3D

De section is een
'echte' 3D-container



Daardoor ligt de blauwe div boven
de rode div (ondanks dat de blauwe
div eerder in de HTML staat)

Html

```
<section>
  <div>blauw</div>
  <div>rood</div>
</section>
```

CSS

```
section {
  perspective(15em);
  transform-style:preserve-3D;
}
```

```
div:nth-of-type(1) {
  background-color:DodgerBlue;
  transform:translateZ(4em);
}
```

naar voren

```
div:nth-of-type(2) {
  background-color:Crimson;
  transform:translateZ(-4em);
}
```

naar achteren

INTRO
8 TRANSITION
TRANSFORMATION

сгг сггггггг #5