

CSS challenge #9

ANIMATIES EN EEN BEEETJE JS INTRO

Animeren recap



[Val Head](#) heeft het interessante boek 'Designing Interface Animation' geschreven, waar ze een aantal **redenen** geeft om animaties in een user interface toe te passen:

- **Relaties tussen elementen leggen**

bijv. een filter dat in- en uitschuift zodat je weet hoe 'de wereld' in elkaar zit en samenhangt.

- **Aandacht op iets vestiging**

bijv. nieuwe content na een refresh met een animatie toevoegen aan een pagina, zodat het even het focuspunt wordt.

- **Feedback geven**

bijv. een winkelwagentje even groter laten worden als er een product wordt toegevoegd, zodat duidelijk is dat het toevoegen gelukt is en waar het product is terug te vinden.

- **Iets uitleggen**

bijv. gebruikers de eerste keer een animatie van een gesture tonen, zodat ze weten hoe de interactie werkt.

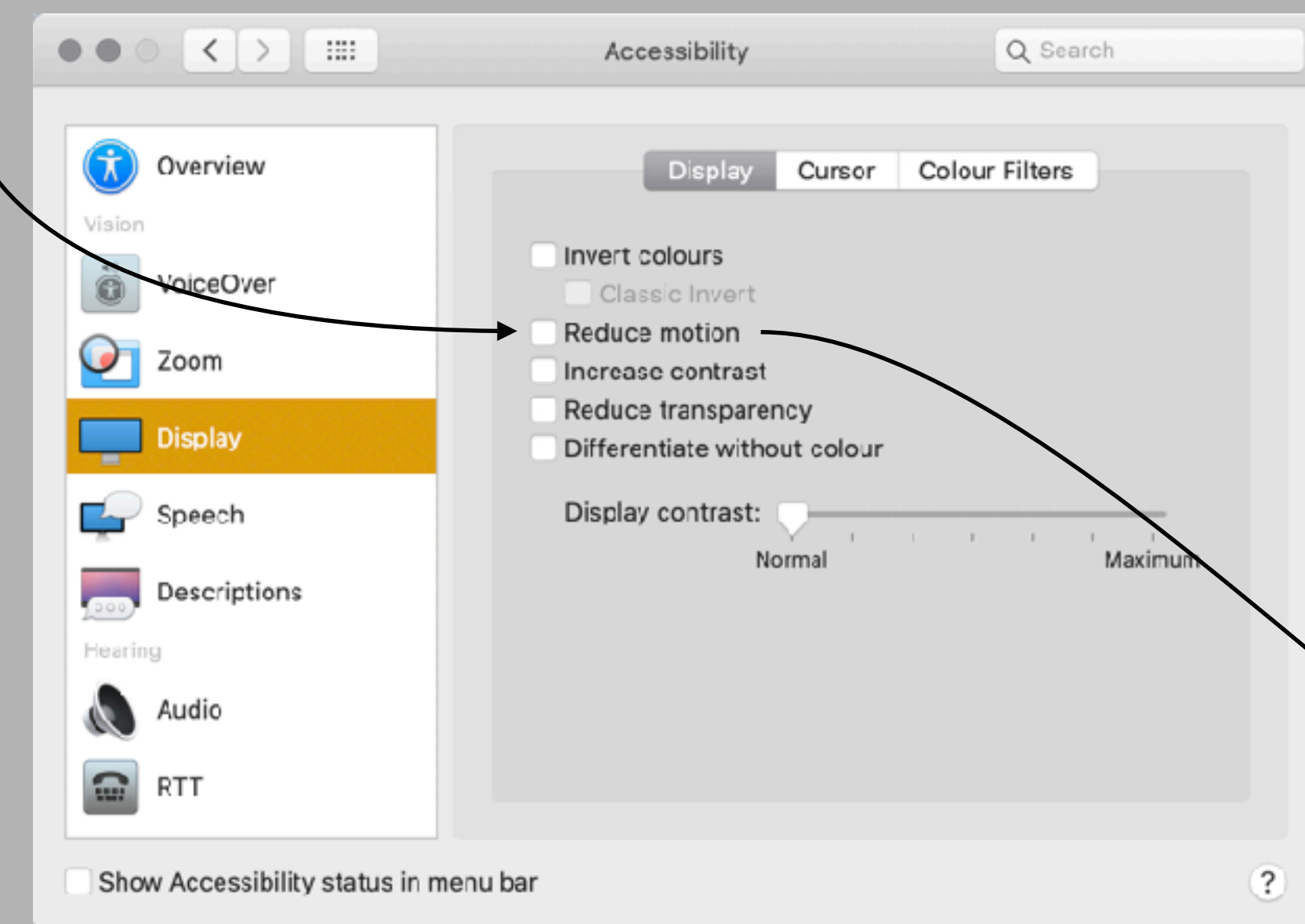
- **Het merk karakter geven**

De branding ook smoel geven met de wijze van animeren.

Pas ook op

Animaties zijn tof en nuttig (zeker als je ze spaarzaam inzet). Maar er zijn ook mensen die misselijk of duizelig van animaties worden of zelfs epileptische aanvallen krijgen.

Gelukkig is er hulp. In de settings van je device kun je de **optie 'Reduce Motion'** aanzetten. Die setting kun je als developer vervolgens [met een media query uitlezen](#) om een variant van je website te maken met verantwoorde of helemaal geen animaties.



Interessant voor de
Zwarte piste om mee
aan de slag te gaan.

Nb. Je ziet **meer opties** waarmee je rekening kunt houden. Die kun je [allemaal uitlezen met media queries](#) (soms nog experimenteel)

Animatie anatomie

[Meet de animation properties \(MDN\)](#)

DEMO



1/4

Animatie anatomie

1. Element bepalen dat gaat animeren

2. Animatie definiëren

3. Animatie aan (status van) element koppelen

4. Duur van de animatie bepalen



De HTML

```
<div>♥</div>
```

De CSS

2a/4

Animatie anatomie

1. Element bepalen dat gaat animeren
- 2. Animatie definiëren**
3. Animatie aan (status van) element koppelen
4. Duur van de animatie bepalen



De tweede stap is de animatie definiëren. Dat doe je met @keyframes. Daarachter staat de naam van de animatie. Die mag je zelf bedenken. Handig om de animatie een betekenisvolle naam te geven, zodat je later de code nog snapt (en anderen ook).

De HTML

```
<div>♥</div>
```

De CSS

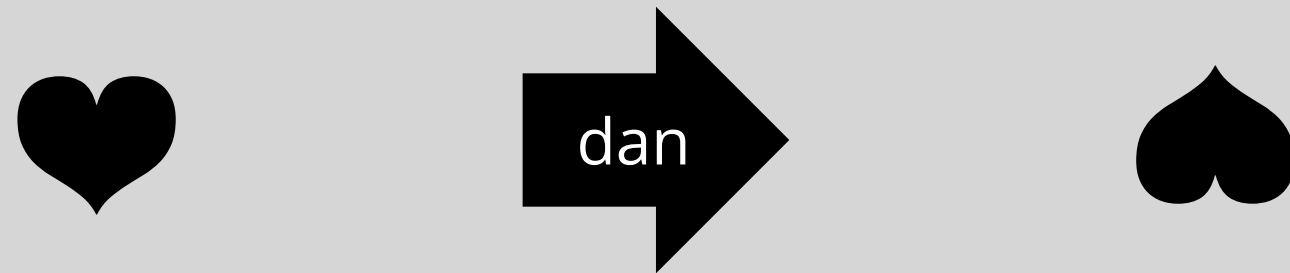
```
@keyframes hartjeDraaien {  
  
}
```

Naam van
de animatie

Animatie anatomie

2b/4

1. Element bepalen dat gaat animeren
- 2. Animatie definiëren**
3. Animatie aan (status van) element koppelen
4. Duur van de animatie bepalen



Start

Vervolgens definieer je het startpunt van de animatie met 0% {}.
Daarbinnen zet je de properties met de waardes die het element bij de start van de animatie moet hebben.

Einde

Dan definieer je het eindpunt van de animatie met 100% {}.
Daarbinnen zet je de properties met de waardes die het element bij het einde van de animatie moet hebben.

In het voorbeeld zal het element 180 graden gedraaid gaan worden.

De HTML

```
<div>♥</div>
```

De CSS

```
@keyframes hartjeDraaien {  
  0% {  
    rotate: 0deg;  
  }  
  100% {  
    rotate: 180deg;  
  }  
}
```

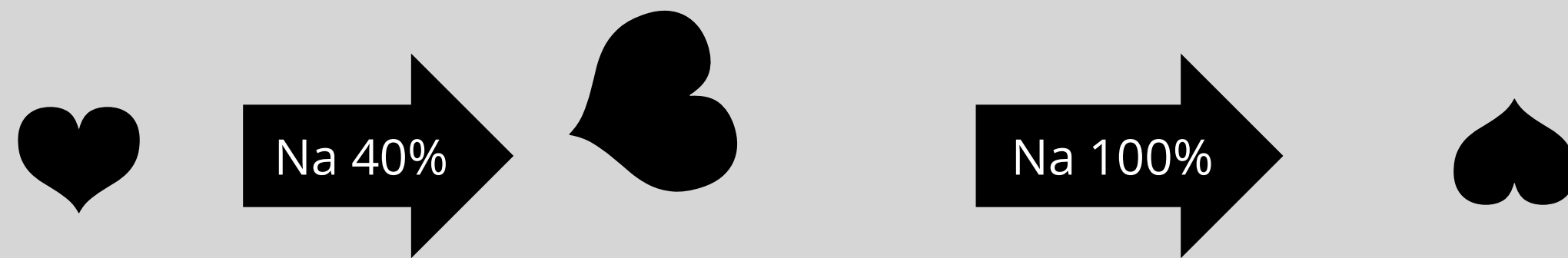
Startpunt van
de animatie

Eindpunt van
de animatie

2c/4

Animatie anatomie

1. Element bepalen dat gaat animeren
- 2. Animatie definiëren**
3. Animatie aan (status van) element koppelen
4. Duur van de animatie bepalen



Extra tussenstops

Een animatie kan uitgebreid worden met meer tussenstops. In het voorbeeld wordt het hartje op 40% van de animatie 2x zo groot om dan verder te draaien en weer de normale grootte aan te nemen.

Nb. Niet alle CSS properties kun je animeren. Kijk voor een lijst met properties die je kunt animeren op: developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties

De HTML

```
<div>♥</div>
```

De CSS

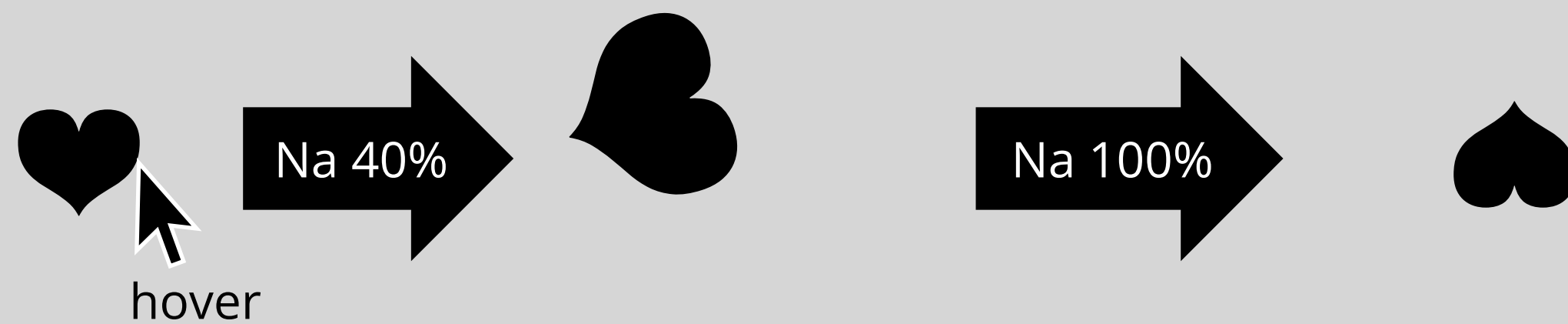
```
@keyframes hartjeDraaien {  
  0% {  
    rotate: 0deg;  
  }  
  40% {  
    scale: 2;  
  }  
  100% {  
    rotate: 180deg;  
  }  
}
```

Extra tussenstop

3/4

Animatie anatomie

1. Element bepalen dat gaat animeren
2. Animatie definiëren
- 3. Animatie aan (status van) element koppelen**
4. Duur van de animatie bepalen



Animation-name

De volgende stap is het koppelen van de animatie aan het element dat moet gaan animeren. Dat doe je door het element (of een status van het element) te selecteren. Om daarbinnen met `animation-name` de animatie te koppelen.

In het voorbeeld wordt de animatie gekoppeld aan de `hover`-status van de `div`. De animatie wordt dus geactiveerd als de gebruiker over de `div` hoovert.

De HTML

```
<div>♥</div>
```

De CSS

```
@keyframes hartjeDraaien {  
  0% {  
    rotate: 0deg;  
  }  
  40% {  
    scale: 2;  
  }  
  100% {  
    rotate: 180deg;  
  }  
}
```

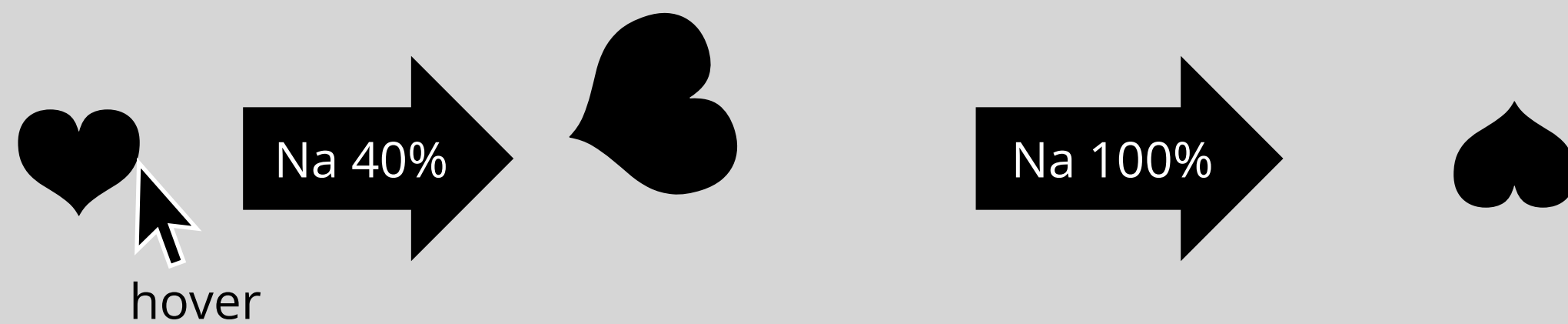
```
div:hover {  
  animation-name: hartjeDraaien;  
}
```

De property
animation-name

4a/4

Animatie anatomie

1. Element bepalen dat gaat animeren
2. Animatie definiëren
3. Animatie aan (status van) element koppelen
- 4. Duur van de animatie bepalen**



Animation-duration

Tenslotte bepaal je hoe lang de animatie gaat duren. Dit doe je met de property `animation-duration`.

In het voorbeeld is de duur 1 seconde. Het hartje zal bij hoveren dus in 1 seconde 180 graden draaien.

De HTML

```
<div>♥</div>
```

De CSS

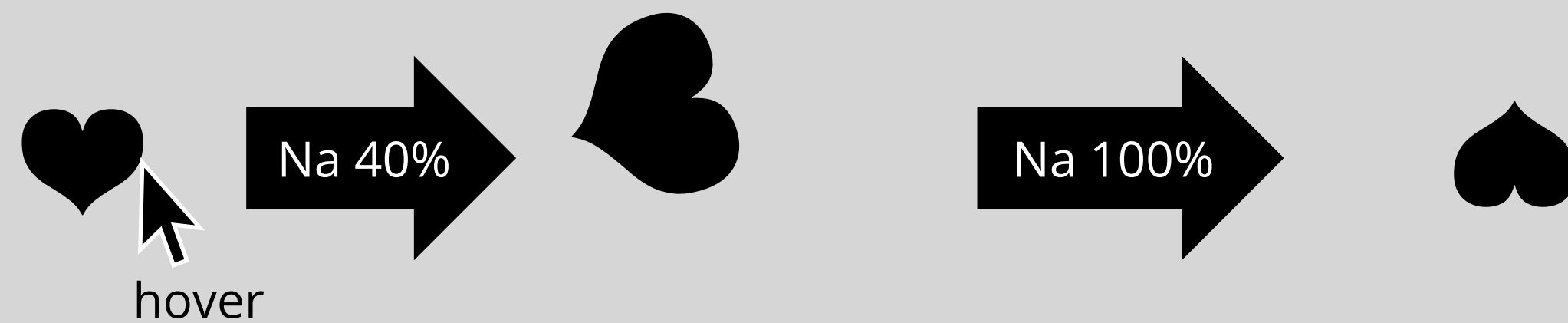
```
@keyframes hartjeDraaien {  
  0% {  
    rotate: 0deg;  
  }  
  40% {  
    scale: 2;  
  }  
  100% {  
    rotate: 180deg;  
  }  
}  
  
div:hover {  
  animation-name: hartjeDraaien;  
  animation-duration: 1s;  
}
```

De property
`animation-duration`

4b/4

Animatie anatomie

1. Element bepalen dat gaat animeren
2. Animatie definiëren
3. Animatie aan (status van) element koppelen
- 4. Duur van de animatie bepalen**



Andere animation properties

Uitleg over meer animation properties: developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations (MDN)

De HTML

```
<div>♥</div>
```

De CSS

```
@keyframes hartjeDraaien {  
  0% {  
    rotate: 0deg;  
  }  
  40% {  
    scale: 2;  
  }  
  100% {  
    rotate: 180deg;  
  }  
}
```

```
div:hover {  
  animation-name: hartjeDraaien;  
  animation-duration: 1s;  
  animation-delay: 1s;  
  animation-timing-function: linear;  
  animation-iteration-count: 10;  
}
```

of bijv:

```
animation-iteration-count: infinite;  
animation-direction: alternate;
```

De animatie start na 1 seconde

Het hartje draait in een tempo (er is geen versnelling) - dat ziet er meestal wat doods uit

De animatie wordt 10x uitgevoerd. Het hartje draait 180 graden - en springt dag weer terug - en dan 10x.

De animatie blijft voor altijd herhalen

De animatie draait heen en draait dan ook weer terug (om-en-om)

prefers-reduced-motion

[houd rekening met de voorkeur van de gebruiker \(MDN\)](#)

Prefers reduced motion

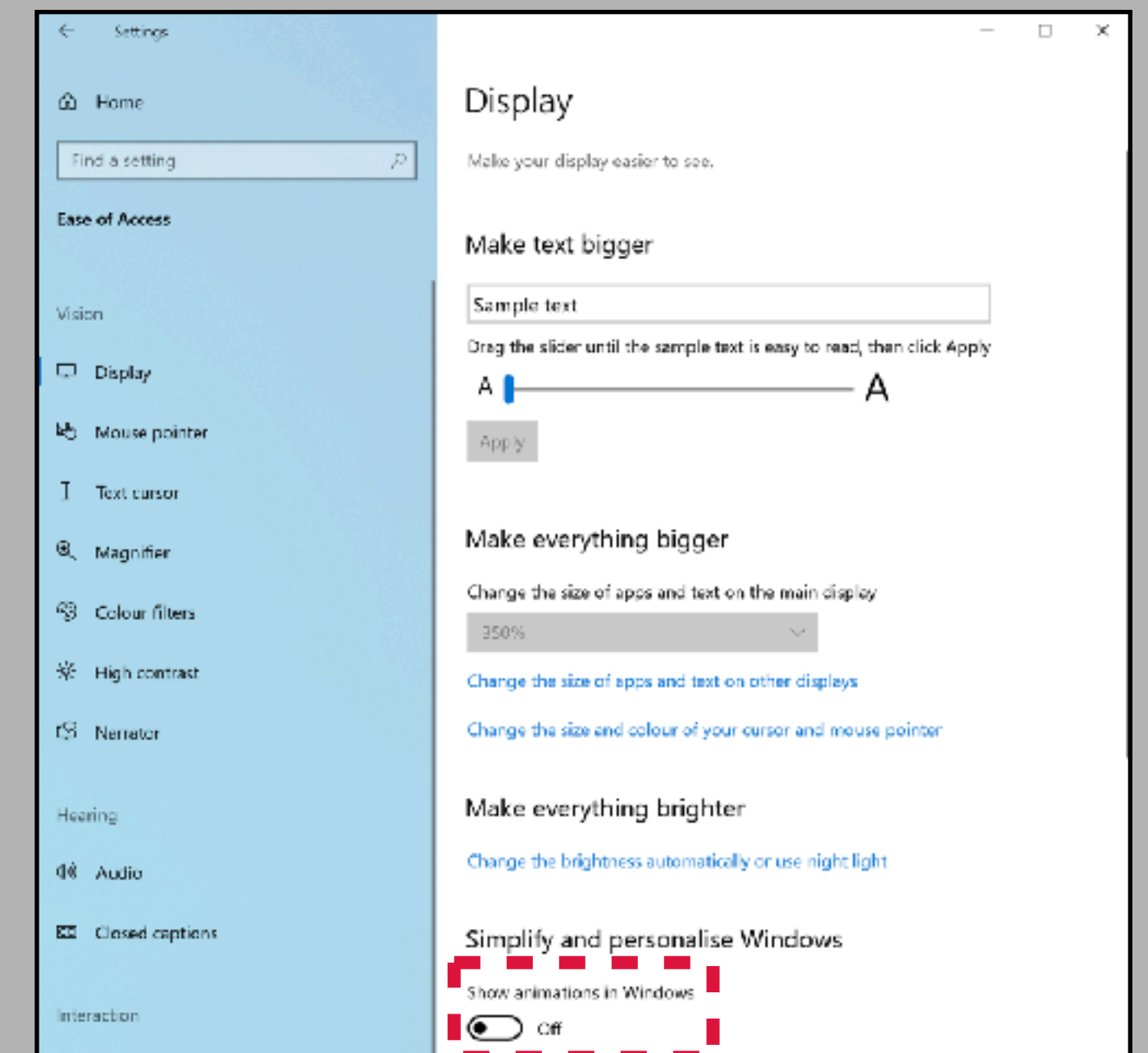
```
@media (prefers-reduced-motion:no-preference) {  
  div {  
    animation-name:go;  
    animation-duration:2s;  
  }  
}
```

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

```
@keyframes go {  
  0% {  
    rotate:0turn;  
  }  
  100% {  
    rotate:10turn;  
  }  
}
```

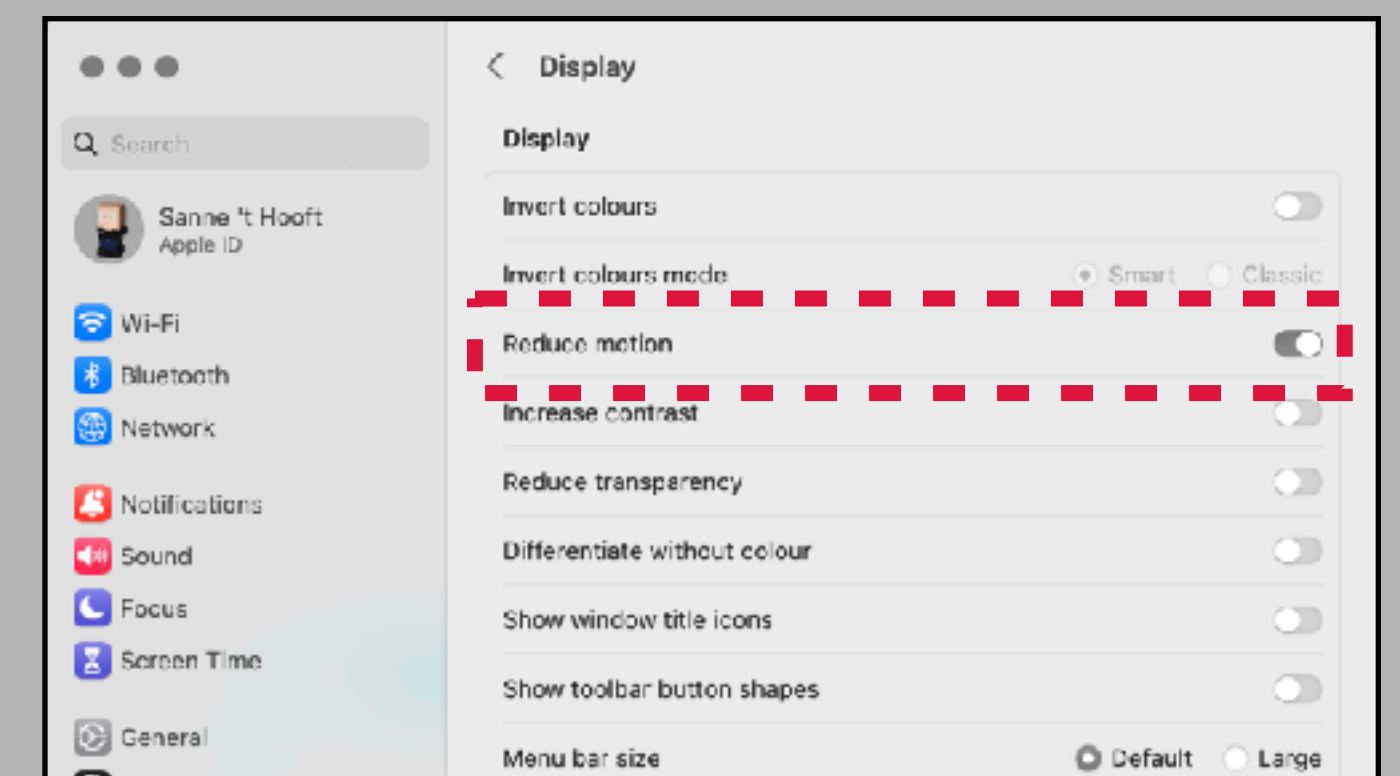
Show animations - Windows

Settings → Ease of Access → Display



Reduce motion

System preferences → Accessibility → Display



Keyframe graph

naar een idee van [amit sheen](#)



Keyframes
graph



0%

50%


100%

Keyframes graph			
			
	0%	50%	100%
translate:			

Keyframes graph			
			
	0%	50%	100%
translate:	0 0	0 -1em	0 0


Keyframes graph			
			
	0%	50%	100%
translate:	0 0	0 -1em	0 0
scale:			

Keyframes graph			
	0%	50%	100%
	translate: 0 0	0 -1em	0 0
	scale: 1 1	2 2	1 1

Keyframes graph			
	0%	50%	100%
	translate: 0 0	0 -1em	0 0
	scale: 1 1	2 2	1 1


```
De code
@keyframes jump (
}

```

Keyframes graph			
	0%	50%	100%
	translate: 0 0	0 -1em	0 0
	scale: 1 1	2 2	1 1


De code

```
@keyframes jump (  
  0% {  
    translate: 0 0;  
    scale: 1;  
  }  
  
)
```

Keyframes graph			
	0%	50%	100%
	translate: 0 0	0 -1em	0 0
	scale: 1 1	2 2	1 1

De code

```
@keyframes jump (  
  
  0% {  
    translate: 0 0;  
    scale: 1;  
  }  
  
  50% {  
    translate: 0 -1em;  
    scale: 2;  
  }  
  
}
```

Keyframes graph			
			
	0%	50%	100%
	translate: 0 0	0 -1em	0 0
scale:	1 1	2 2	1 1

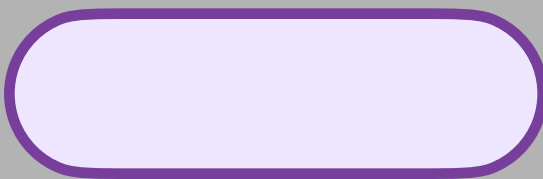
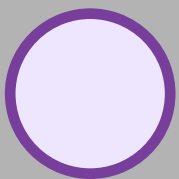
De code

```
@keyframes jump (  
  
  0% {  
    translate: 0 0;  
    scale: 1;  
  }  
  
  50% {  
    translate: 0 -1em;  
    scale: 2;  
  }  
  
  100% {  
    translate: 0 0;  
    scale: 1;  
  }  
}
```

Keyframes
graph



3D rotate
on y-as



Why did the
computer catch
cold? It left a
window open.

0%

33%

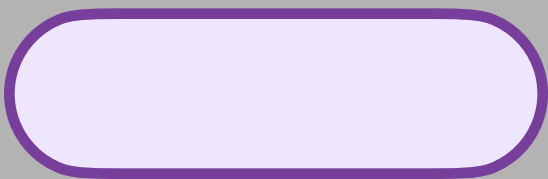
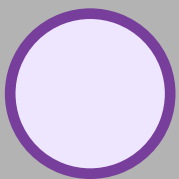
67%

100%

Keyframes
graph



3D rotate
om y-as



Why did the
computer catch
cold? It left a
window open.

0%

33%

67%

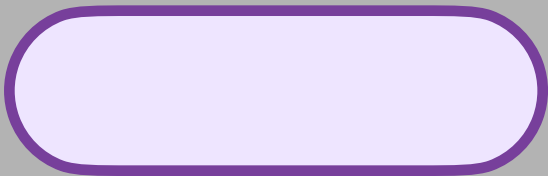
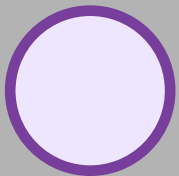
100%

width:

Keyframes
graph



3D rotate
on y-as



Why did the
computer catch
cold? It left a
window open.

0%

33%

67%

100%

width:

2.5em

2.5em

11em

11em

Keyframes
graph



3D rotate
on y-as



Why did the
computer catch
cold? It left a
window open.

0%

33%

67%

100%

width:

2.5em

2.5em

11em

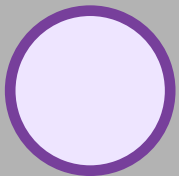
11em

height:

Keyframes
graph



3D rotate
on y-as



Why did the
computer catch
cold? It left a
window open.

0%

33%

67%

100%

width:

2.5em

2.5em

11em

11em

height:

2.5em

2.5em

2.5em

11em

Keyframes
graph



3D rotate
on y-as



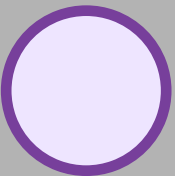
Why did the
computer catch
cold? It left a
window open.

	0%	33%	67%	100%
width:	2.5em	2.5em	11em	11em
height:	2.5em	2.5em	2.5em	11em
background:				

Keyframes
graph

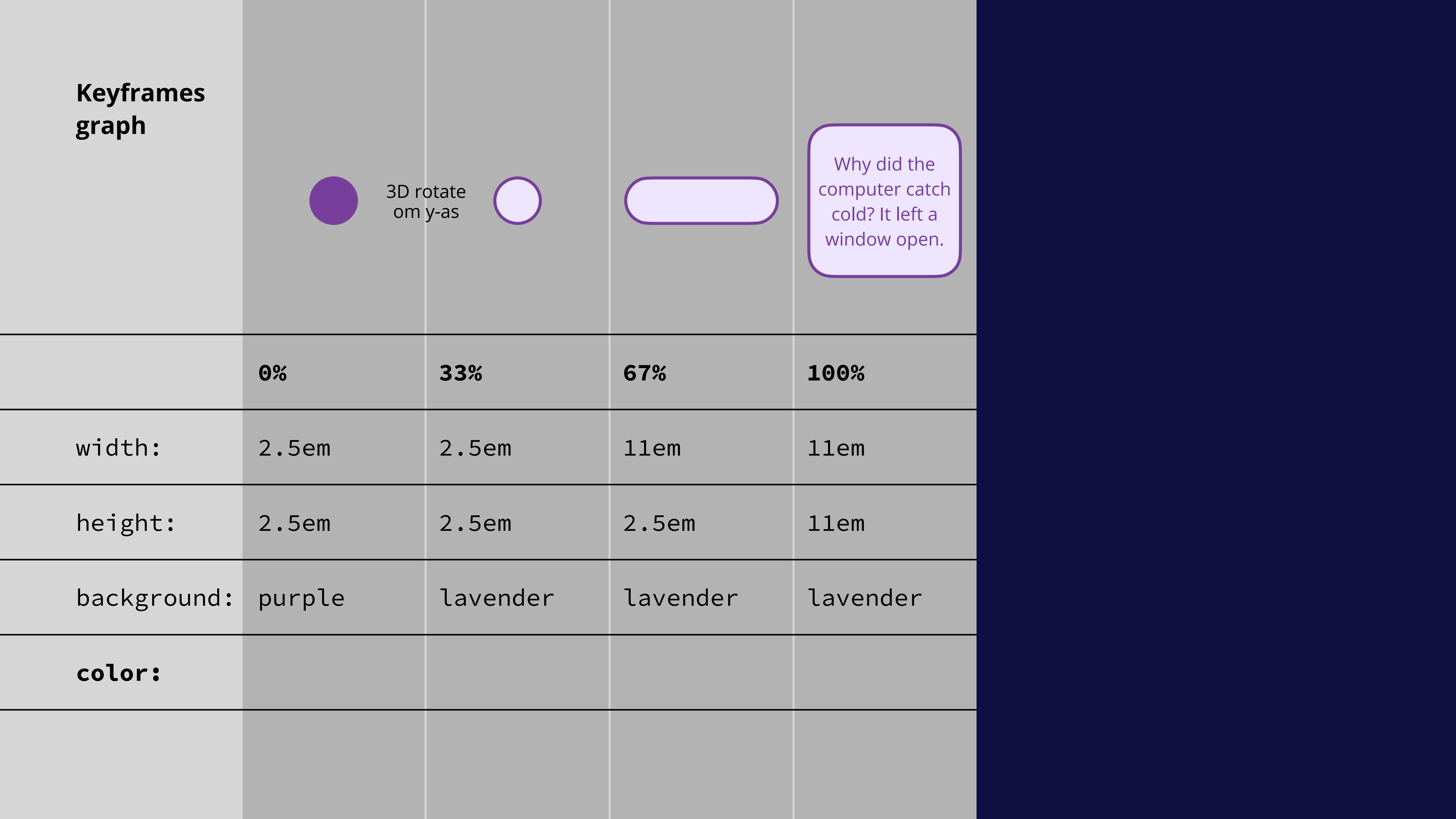


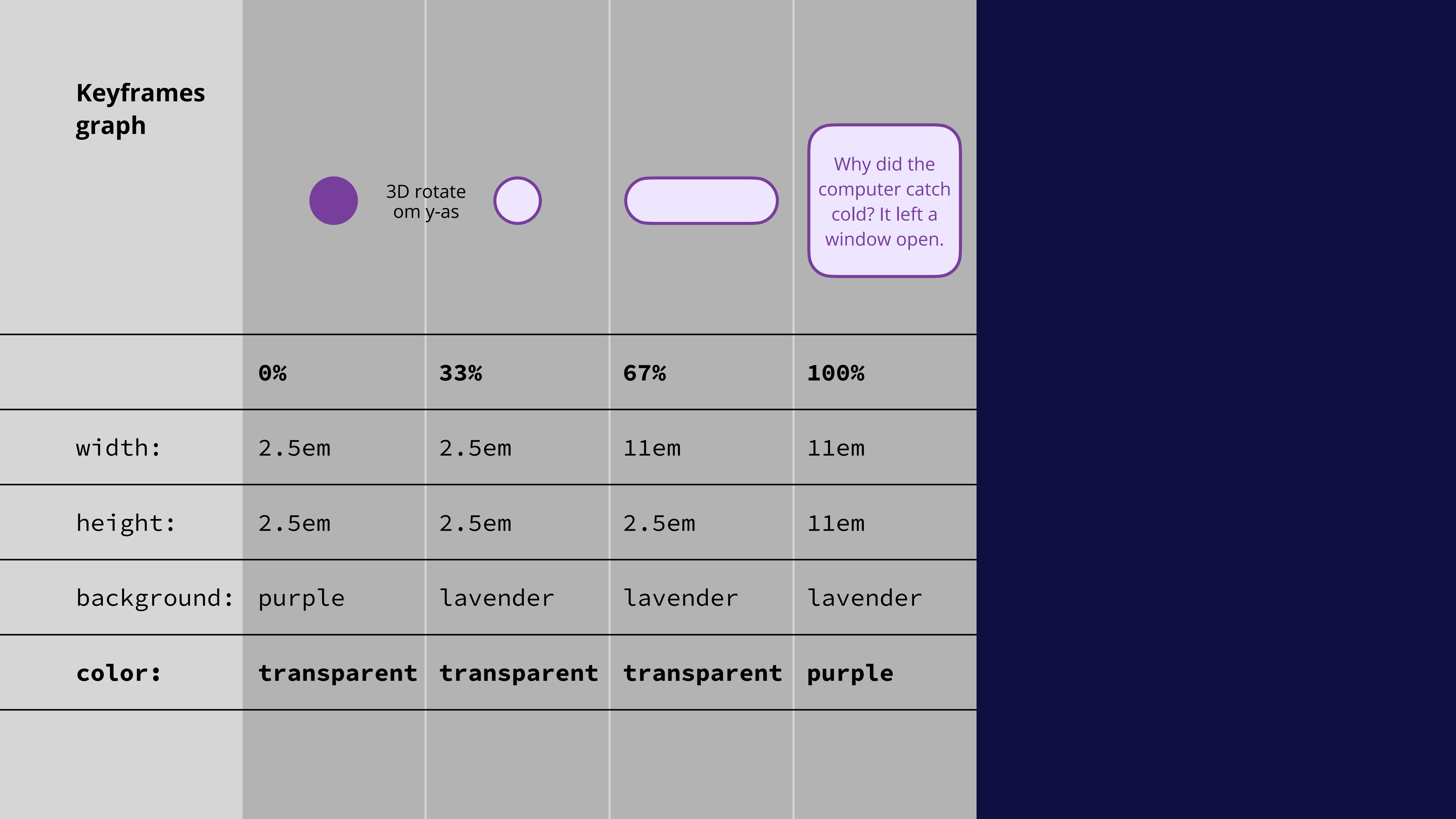
3D rotate
on y-as



Why did the
computer catch
cold? It left a
window open.


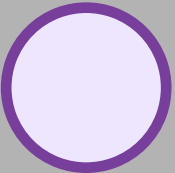

	0%	33%	67%	100%
width:	2.5em	2.5em	11em	11em
height:	2.5em	2.5em	2.5em	11em
background:	purple	lavender	lavender	lavender





Keyframes graph	<div><div><div></div><div>3D rotate om y-as</div><div></div><div></div><div>Why did the computer catch cold? It left a window open.</div></div></div>				
	0%	33%	67%	100%	
	width:	2.5em	2.5em	11em	11em
	height:	2.5em	2.5em	2.5em	11em
	background:	purple	lavender	lavender	lavender
color:	transparent	transparent	transparent	purple	
rotate:					

Keyframes graph	<div><div><div></div><div>3D rotate om y-as</div><div></div><div></div><div></div></div></div>				
	0%	33%	67%	100%	
	width:	2.5em	2.5em	11em	11em
	height:	2.5em	2.5em	2.5em	11em
	background:	purple	lavender	lavender	lavender
color:	transparent	transparent	transparent	purple	
rotate:	y -180deg	y 0deg	y 0deg	y 0deg	

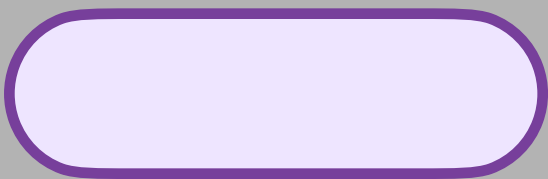
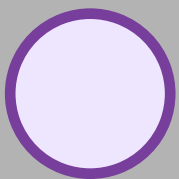
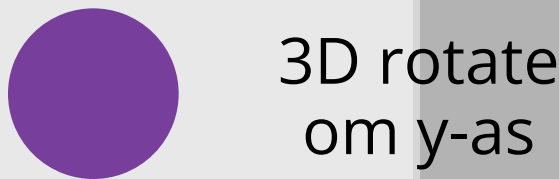
Keyframes graph	<div> <div>  <div>3D rotate om y-as</div>  </div>  <div> <div>Why did the computer catch cold? It left a window open.</div> </div> </div>			
	0%	33%	67%	100%
width:	2.5em	2.5em	11em	11em
height:	2.5em	2.5em	2.5em	11em
background:	purple	lavender	lavender	lavender
color:	transparent	transparent	transparent	purple
rotate:	y -180deg	y 0deg	y 0deg	y 0deg

De code

```
@keyframes grow {
```

```
}
```

Keyframes
graph



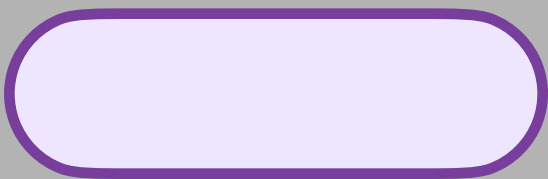
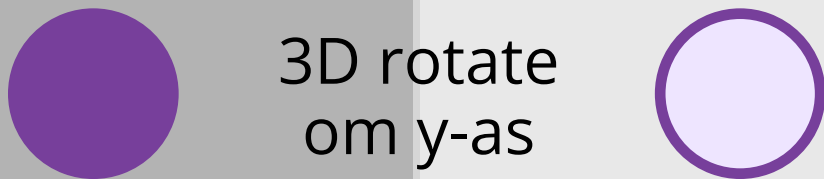
Why did the
computer catch
cold? It left a
window open.

	0%	33%	67%	100%
width:	2.5em	2.5em	11em	11em
height:	2.5em	2.5em	2.5em	11em
background:	purple	lavender	lavender	lavender
color:	transparent	transparent	transparent	purple
rotate:	y -180deg	y 0deg	y 0deg	y 0deg

De code

```
@keyframes grow (  
  
  0% {  
    width: 2.5em;  
    height: 2.5em;  
    background:purple;  
    color: transparent;  
    rotate: y -180deg;  
  }  
  
)
```

Keyframes
graph



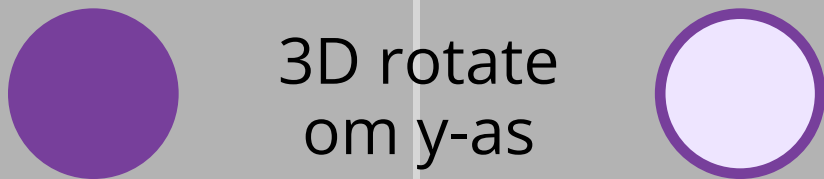
Why did the
computer catch
cold? It left a
window open.

	0%	33%	67%	100%
width:	2.5em	2.5em	11em	11em
height:	2.5em	2.5em	2.5em	11em
background:	purple	lavender	lavender	lavender
color:	transparent	transparent	transparent	purple
rotate:	y -180deg	y 0deg	y 0deg	y 0deg

De code

```
@keyframes grow (  
  
  0% {  
    width: 2.5em;  
    height: 2.5em;  
    background:purple;  
    color: transparent;  
    rotate: y -180deg;  
  }  
  
  33% {  
    width: 2.5em;  
    background:lavender;  
    rotate: y 0deg;  
  }  
  
}
```

Keyframes
graph



Why did the
computer catch
cold? It left a
window open.

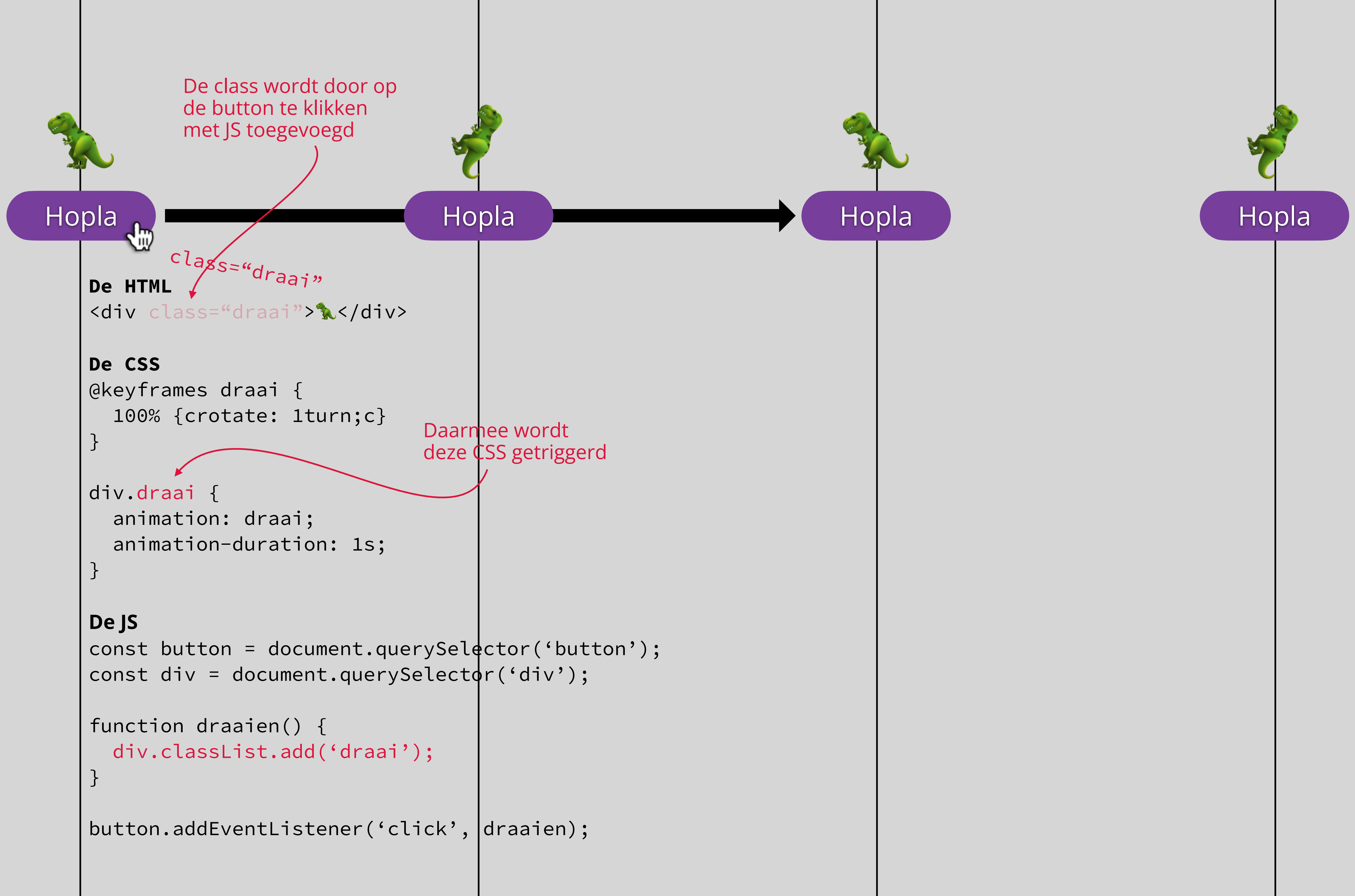
	0%	33%	67%	100%
width:	2.5em	2.5em	11em	11em
height:	2.5em	2.5em	2.5em	11em
background:	purple	lavender	lavender	lavender
color:	transparent	transparent	transparent	purple
rotate:	y -180deg	y 0deg	y 0deg	y 0deg

De code

```
@keyframes grow (  
  
  0% {  
    width: 2.5em;  
    height: 2.5em;  
    background:purple;  
    color: transparent;  
    rotate: y -180deg;  
  }  
  
  33% {  
    width: 2.5em;  
    background:lavender;  
    rotate: y 0deg;  
  }  
  
  67% {  
    width: 11em;  
    height: 2.5em;  
    color: transparent;  
  }  
  
}
```

JS animation events

[animantionstart](#), [animantioniteration](#), [animationcancel](#) en [animantionend](#)





Hopla



Hopla



Hopla

De class staat er na het draaien nog steeds



De HTML

```
<div class="draai">🦖</div>
```

De CSS

```
@keyframes draai {  
  100% {transform: rotate(1turn);}  
}
```

```
div.draai {  
  animation: draai;  
  animation-duration: 1s;  
}
```

De JS

```
const button = document.querySelector('button');  
const div = document.querySelector('div');
```

```
function draaien() {  
  div.classList.add('draai');  
}
```

```
button.addEventListener('click', draaien);
```

Daardoor kan de animatie door te klikken niet nogmaals getriggerd worden :-)



Hopla



Hopla



Hopla



De class wordt na het einde van de animatie weer verwijderd :-)



Hopla

De HTML

```
<div class="draai">
```

De JS

```
const button = document.querySelector('button');  
const div = document.querySelector('div');
```

```
function draaien() {  
  div.classList.add('draai');
```

```
  div.addEventListener('animationend', reset);  
}
```

```
function reset() {  
  div.classList.remove('draai');  
}
```

```
button.addEventListener('click', draaien);
```

De div geeft aan
wanneer deze klaar
is met draaien

Dan wordt de reset
function uitgevoerd.
Daarin wordt de class
van de div verwijderd



Hopla



Hopla



Hopla



De class wordt door op de button te klikken met JS toegevoegd

class="draai"

De HTML

```
<div class="draai">🦖</div>
```

De CSS

```
@keyframes draai {  
  100% {transform: rotate(1turn);}  
}
```

Daarmee wordt deze CSS weer getriggerd

```
div.draai {  
  animation: draai;  
  animation-duration: 1s;  
}
```

De JS

```
const button = document.querySelector('button');  
const div = document.querySelector('div');
```

De class kan nu weer toegevoegd worden. Daarmee kan de animatie weer getrieerd worden.

```
function draaien() {  
  div.classList.add('draai');
```

```
  div.addEventListener('animationend', reset);  
}
```

```
function reset() {
```



Hopla



WILBO
BEEBLE
EIE EEEI
ANIMALIES

css challenge #2