

Css3 transition og animation – tid og bevægelse

Css3 rummer to muligheder for at opbygge css-regler der forandrer hvordan et html-elements grafiske fremtræden som en formmæssig forandring, der afvikles animeret over tid, nemlig **transition** og **animation**:

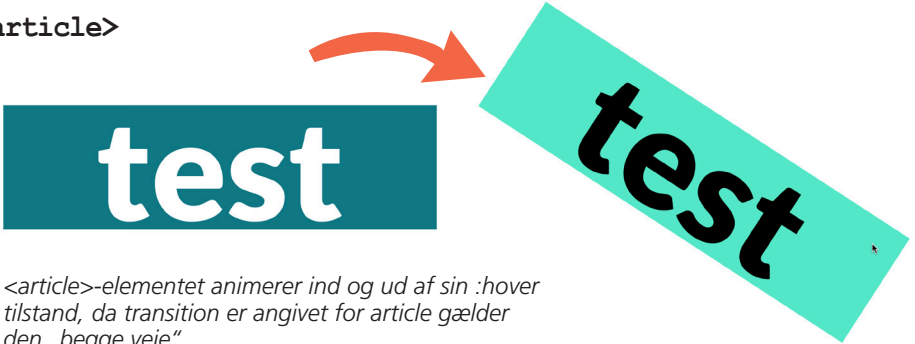
1 Transition og :hover

En **transition** er en »overgang« mellem to css-styrede tilstande. Grundlæggende fungerer det ved at et element tildeles transition-„tid“, hvorefter css-ændringer vil afvikles som animerede „overgange“

Skal du teste en transition – uden brug af scriptbaseret „logik“ – kræver det at et element får nye ændrede css-egenskaber, som resultat af brugerinteraktion. Her er der det mest oplagte brug af pseudoclassen :hover...

```
<article><h1>test</h1></article>
```

```
article {  
  width:700px;  
  height:200px;  
  background: #1a7984;  
  color:white;  
  transition:1000ms;  
  margin:0 auto; }  
  
article:hover {  
  transform:rotate(33deg);  
  background:#5ee8ca;  
  color:black;}  
  
h1 {  
  font-weight: 600;  
  font-size: 14rem;  
  text-align: center;  
  line-height: 200px;}
```



<article>-elementet animerer ind og ud af sin :hover tilstand, da transition er angivet for article gælder den „begge veje“.

Ved også at angive en transition-tid i article:hover kan de to animationer afvikles med hver sin hastighed:

```
article {  
  ....  
  transition:1000ms; }  
  
article:hover {  
  .....  
  transition:300ms; }
```

Transition-tid fra :hover-tilstand

Transition-tid til :hover-tilstand

Forskellig transition-hastighed på specifikke css-egenskaber

I stedet for at have samme hastighed på alle css-ændringer, kan separate egenskaber tildeles deres egen tid:

```
article {  
  width:700px;  
  height:200px;  
  background: #1a7984;  
  color:white;  
  transition:transform 1000ms, color 500ms, background 1.4s;  
  margin:0 auto; }
```

Tid kan angives i milisekunder og sekunder

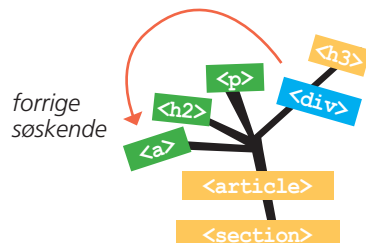
Ud over tid, kan alle ændringer af css-egenskaber forsynes med forsinkelse (delay) og acceleration (easing) – se mere lidt senere!

2

Fjernstyring af andet elements transition med :hover og general sibling

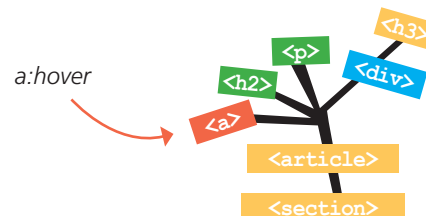
Ved :hover pseudoclass kan du overføre css og hermed transitions på det element du interagerer med, men det kan give problemer hvis elementet skalerer eller roterer. For så snart elementet fjerner sig fra musens cursor, som følge af sin animation, er det ikke længere i :hover-tilstand og vil følgelig starte på at animere „baglæns“ til sin normale tilstand. Det bliver noget rod, men heldigvis kan du bruge css3 selektoren „general sibling“.

Med general sibling er det nemlig muligt at få ét elements :hover til at animere et andet søskende-element, se her:



<a>, <h2>, <p> og <div> er søskende. Set fra <div>'s synsvinkel har det en foregående søskende der er et a-tag. Dét kan man udtrykke med denne css3 selektor:

a ~ div



Hvis <a> går i :hover-tilstand pga en cursor der svæver over det, har <div> lige pludselig en tidligere søskende som er et a:hover – og med denne general sibling-selektor kan vi igangsætte en transition:

a:hover ~ div

Oversat til noget konkret: en „slide-in“ box

I eksemplets html-kode rummer <article> et a-tag (der er udsmykket med et svg-billede fra Illustrator), en overskrift (<h2>) samt en brødtekst (<p>) og til slut en <div>. Div'en er absolut positioneret – forskudt ud til venstre for article og har præcist dennes størrelse. Når vi ikke kan se div'en er det fordi article med css fungerer som „maske“ med egenskaben „overflow:hidden“.



Den omtalte og afgørende css:

```
article {
  ...
  position: relative;
  overflow: hidden; }
```

```
article a {
  display: block;
  ...
  background: url(billeder/questionmark.svg) ; }
```

```
div {
  width : 550px;
  position: absolute;
  left: -550px;
  top: 0;
  height: 750px;
  transition: 500ms; }
```

Css-reglen der starter slide-ind boxens animation

```
a:hover ~ div {
  -webkit-transform: translateX(400px); }
```

3

Transition af et :hover-elementets efterkommere eller child-elementer

Man kan opbygge interessante effekter ved at lade elementer, der er efterkommere af et element i :hover-tilstand få aktiveret deres egne transitions. Her kan man fx udstyre de enkelte elementer deres egne animerede transforms (fx flytte sig og rotere i hver sin retning) eller forskyde animationernes start med et individuelt „delay“...

For at ramme elementerne med css-selektorer kan du fx bruge inheritance (fx ul:hover li), child-selektorer (article > div).

Til at udvælge individuelle elementer i søskende-flokken kan du bruge forskellige former for selektorer, fx first-child, last-child og nth-child (li:first-child, li:nth-child(2), li:last-child) eller adjacent sibling (li + li).

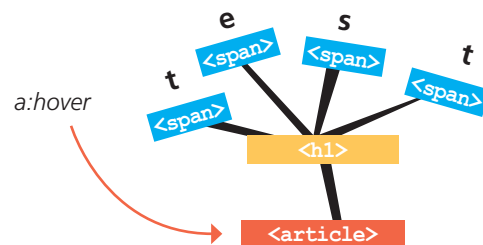
test

test

test

test

```
<article>
<h1>
<span>t</span><span>e</span><span>s</span><span>t</span>
</h1>
</article>
```



Når <article> går i :hover-tilstand starter en transition for den – og dens efterkommer (<h1> og). Teksten i <h1> er opsplittet i ét -tag pr. bogstav og hvert tildeles sin egen transition med selektorerne:

```
article:hover span:first-child
article:hover span+span
article:hover span:nth-child(3)
article:hover span:last-child
```

Eksemplets css-kode

Hvert bogstav „arver“ <article>’s animation når den indtræder i sin :hover-tilstand, idet de roterer sammen med <h1> – men de får også deres helt egen individuelle animation med disse css-regler:

```
article span { display:inline-block; transition:1s; color:white;}

article:hover span {
  transform:scale(1.2,1.2) translate(-2rem,-3rem) ; color:#000;}

article:hover span + span {
  transform:scale(2,2) translate(.2rem,2rem) ;}

article:hover span:nth-child(3) {
  transform:scale(1.2,1.2) translate(1rem,-4rem) ;}

article:hover span:last-child {
  transform:scale(1.2,1.2) translate(3rem,2rem) ;}
```

Oversigt: transition egenskaber

I forbindelse med transition er der disse egenskaber:

transition

Tildeler transition til et element, sammen med varighed for overgangseffekten:

-webkit-transition: opacity 2s;
transition: opacity 2s;

Varighed kan angives som sekunder (2s) eller millisekunder (2000ms)
Kan efterhånden bruges uden prefixes

Skal de forskellige animerede egenskaber have hver sin varighed skrives dette pr. egenskab, adskilt af et komma:

-webkit-transition: opacity 2s, color 1s, -webkit-transform 800ms ;
transition: opacity 2s, color 1s, -webkit-transform 800ms;

transition-delay

Indsætter en forsinkelse for animationen:

-webkit-transition-delay: 1s;
transition-delay: 1s;

Transition-delay kan bygges direkte i transition-egenskaben, som et tal nummer to – efter animationens varighed – det kan også gøres pr. egenskab:

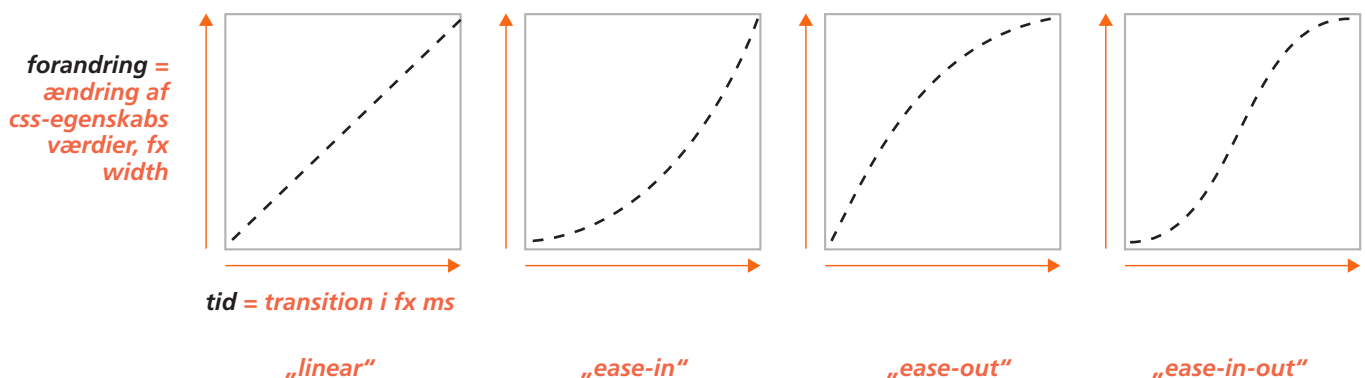
-webkit-transition: opacity 2s 1s, color 1s, -webkit-transform 800ms 1.2s ;
transition: opacity 2s 1s, color 1s, -webkit-transform 800ms 1.2s;

transition-timing-function (– også kendt som „easing“)

Transition-timing styrer acceleration og de-acceleration af animationen, i de fleste animationssammenhænge kalder man det for „easing“. Easing kan kodes separat eller indbygges sammen med tid og delay for de enkelte egenskaber:

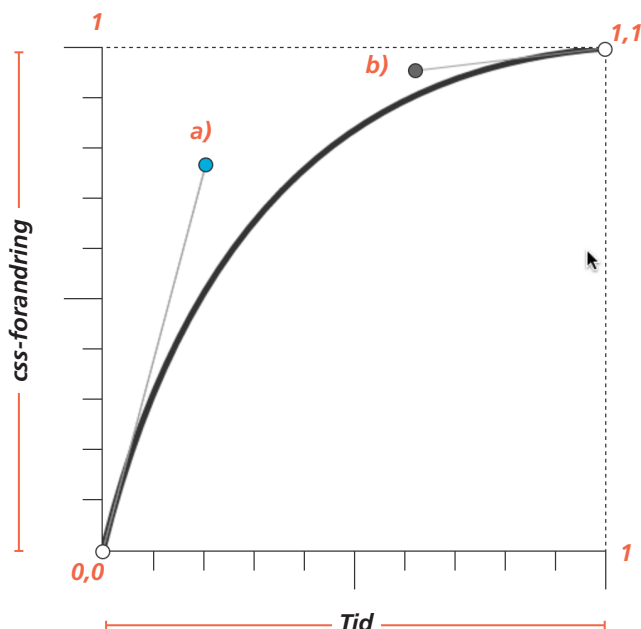
transition-timing-function: ease-in;
eller:
transition: opacity 2s 1s linear, color 1s, -webkit-transform 800ms 1.2s ease-in;

Css3 transition har en række forudindstillede timings (fx „ease-in“), der er lettest at illustrere som en graf:



Kontrollér easing-kurver med Cubic bezier – forklaring

Som grafiker er du fortrolig med at brugen pen-værktøj i AI eller PSD, så du vil nok have nemt ved at „tegne“ dine egne easing-hastighedskurver som cubic bezier-kurver opbygget mellem to ankerpunkter. Dét du angiver for at forme din Cubic bezier-kurve er koordinaterne for kursens styrelinjer, **a)** og **b)**, – mens start-punkt og slutpunkterne altid er givet på forhånd (med koordinaterne **0,0** og **1,1**):



Punktet **a)** befinder sig vandret i værdien 0.2 og lodret omkring 0.78 og får vel koordinaterne „**0.2,0.78**“. Punktet **b)** har cirka koordinaterne **0.62,0.96**

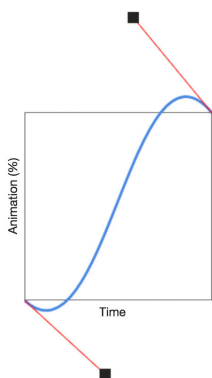
css-reglen cubic-bezier()

Cubic bezier kan indgå i css'en ligesom andre easings:

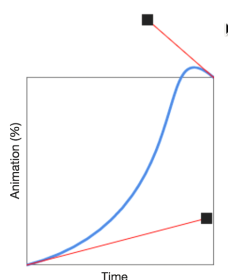
transition-timing-function: cubic-bezier(.20, .77, .62, .96);

eller:

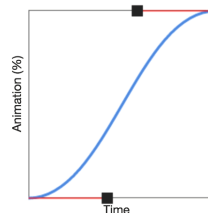
transition: opacity 2s 1s cubic-bezier(.20, .77, .62, .96), color 1s .5s linear;



(0.430, -0.395, 0.580, 1.510);



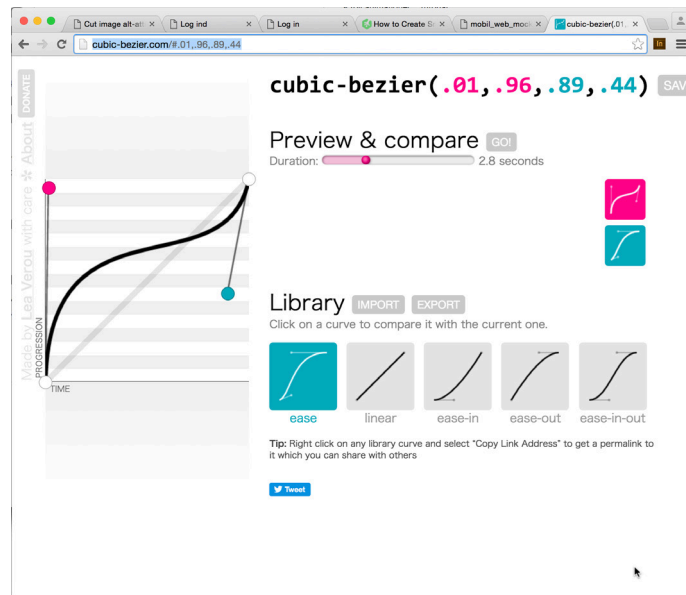
(0.960, 0.250, 0.645, 1.310)



(0.420, 0.000, 0.580, 1.000)

Online cubic-bezier generator

Synes du det er svært at regne cubic-bezier ud, kan du ty til denne generator på nettet, hvor du bare trækker i håndtagene for at få koden. Her kan resultatet oveni købet previewes:



Adressen er: <http://cubic-bezier.com>

Css3 animation og @keyframes

css3 animations kan tildeles elementer præcist som ved transitions, så de strategier der tidligere er beskrevet gælder også her.

Der er dog afgørende forskelle på hvordan transitions og animations opfører sig:

En **transition** har kun to tilstande og overgangen afspilles så længe elementet er genstand for forandring, fx ved :hover. Ophører fx :hover tilstanden kan en transition animere „baglæns“ tilbage til elementets designmæssige udgangspunkt. Det er smukt, og giver elementerne en „dynamisk“ adfærd, der reagerer på brugerens interaktion.

En **animation** kan have mange keyframes der rummer mange forandringer. De kan triggres af fx :hover, men ophører :hover-tilstanden collapse animationen brat til elementets designmæssige udgangspunkt. Det er ikke specielt smukt og opleves ofte som en mærkelig fejl.

Omvendt kan animations afvikles af sig selv uden bruger-interaktion og igangsættes efter fx et delay.

@keyframes

Css3 animation består af to dele, på den ene side en css-blok, **@keyframes { }**, hvor animationens tidsforløb og dens css-ændringer beskrives i kodeform. På den anden side egenskaben **animation**, hvor selve animationens varighed, retning, antal loops etc. defineres.

1 @keyframes – „from“ og „to“

@keyframes beskriver tidsforløbet og de css-forandringer der udgør elementets bevægelse. Tidsforløbet skal navngives og kan beskrives som fra start til slut, sådan her:

```
@-webkit-keyframes roter {  
  from {  
    -webkit-transform: rotate(0deg);  
  }  
  to {  
    -webkit-transform: rotate(-360deg);  
  }  
}
```

I stedet for „from“ og „to“ kan der skrives „0%“ og „100%“

Hvis animationen skal tage udgangspunkt i elementets css-egenskaber før animationen afvikles kan man nøjes med at skrive „to“:

```
@-webkit-keyframes roter {  
  to {  
    -webkit-transform: rotate(-360deg);  
  }  
}
```



3 @keyframes – tid i %

Ved at angive @keyframes i % kan animationens tidsforløb og bevægelse formgives nuanceret med flere css-styrede forandringer. Keyframenes »tidspunkter« angives som % af den samlede tid – 50% er altså halvvejs i animationens tidsforløb:

```
@-webkit-keyframes flyt_fremad {  
  0% {-webkit-transform: translate(170px,0px) rotate(0deg)scale(.5); z-index:1 }  
  10% { z-index:1}  
  11% {-webkit-transform: translate(350px,100px) rotate(30deg); z-index:2}  
  21%{-webkit-transform: translate(30px,320px) rotate(-30deg);}  
  80% {opacity:1.0;}  
  93% {-webkit-transform: translate(85px,320px) rotate(20deg) scale(2);opacity:.5; }  
  100% {-webkit-transform: translate(30px,320px) rotate(20deg) scale(10);opacity:0; z-index:2}  
}
```

Tiden angives her i % og der er mange keyframes, der ikke nødvendigvis alle behøver at ændre samtlige involverede css-egenskaber

4 Pauser i @keyframes

Du kan indbygge forsinkelser i keyframes ved at gentage samme css-egenskaber, så der „sker“ det samme på to efterfølgende keyframes:

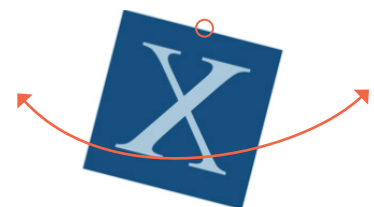
```
@keyframes vandret {  
  0% { -webkit-transform: translateX(0); }  
  50% { -webkit-transform: translateX(350px);}  
  70% { -webkit-transform: translateX(350px);}  
  100% { -webkit-transform: translateX(700px);}  
}
```

Der bliver her et kort ophold mellem **keyframe 50%** og **70%** for der sker ingen ændringer i css-egenskaberne

5 Easing defineret i @keyframes

Du kan opbygge easings i keyframes ved selv at skabe css-forandringer på de rette tidspunkter. Det kræver lidt øvelse:

```
@keyframes swing {  
  10% {-webkit-transform: rotateZ(-15deg); }  
  20% {-webkit-transform: rotateZ(15deg); }  
  30% {-webkit-transform: rotateZ(-10deg); }  
  40% {-webkit-transform: rotateZ(5deg); }  
  50% {-webkit-transform: rotateZ(-5deg); }  
  60% {-webkit-transform: rotateZ(0deg); }  
}
```



Rotationen bliver mindre og mindre for hver keyframe, indtil X'et falder til ro...



Oversigt: animation egenskaber

For at skabe en animation for et element skal de forud-definerede keyframes tilknyttes en css **animation** regel for det element, der skal animere – fx sådan her:

```
@keyframes swing {  
  10% { -webkit-transform: rotateZ(-15deg); }  
  20% { -webkit-transform: rotateZ(15deg); }  
  30% { -webkit-transform: rotateZ(-10deg); }  
  40% { -webkit-transform: rotateZ(5deg); }  
  50% { -webkit-transform: rotateZ(-5deg); }  
  60% { -webkit-transform: rotateZ(0deg); }  
}  
  
#box {  
  -webkit-animation: swing 1s;  
  animation: swing 1s;  
}
```

Definition af @keyframes med navnet „swing“

Oprettelse af animation, hvor elementet tilknyttes animationens varighed og de foruddefinerede keyframes

animation-name og animation-duration

Tildeler et element animation i form af et sæt navngivne keyframes og angiver animationens samlede varighed:

```
#box {  
  -webkit-animation-name: animer_keyframes;  
  animation-name: animer_keyframes;  
  -webkit-animation-duration: 4s;  
  animation-duration: 4s; }  
}
```

Eller kortere:

```
#box {  
  -webkit-animation: animer_keyframes 4s;  
  animation: animer_keyframes 4s; }  
}
```

animation-iteration-count

Angiver hvor mange gange afspilningen skal loope, eller køre uendeligt („infinite“):

```
#box {  
  -webkit-animation-count: 10;  
  animation-count: 10; }  
}
```

Eller kortere:

```
#box {  
  -webkit-animation: animer_keyframes 4s 10;  
  animation: animer_keyframes 4s 10; }  
}
```

animation-fill-mode

Angiver hvordan elementet skal efterlades efter at animation har afviklet sidste frame. Som default vil det vende tilbage til starten af animationen, men værdien „forwards“ stopper animationen og fryser hermed elementets animerede forandring i sidste frame.

```
#box {  
  -webkit-animation-fill-mode: forwards;  
  animation-fill-mode: forwards }  
}
```

Eller kortere:

```
#box {  
  -webkit-animation: animer_keyframes 4s forwards;  
  animation: nimer_keyframes 4s forwards; }  
}
```

animation-direction

Animation-direction styrer om animationen skal afspilles forlæn, baglæns eller skifte mellem forlæns og baglæns („alternate“), skifte mellem baglæns og forlæns, men med første afspilning baglæns („alternate-reverse“). Almindelig forlæns afspilning har værdien „normal“. Det er værd at bemærke at der også skal angive et antal loops på mindst „2“ før alternate vil virke:

```
#box {  
  -webkit-animation-direction: alternate;  
  animation-direction: alternate }  
}
```

Eller kortere:

```
#box {  
  -webkit-animation: animer_keyframes 4s 2 alternate;  
  animation: nimer_keyframes 4s 2 alternate; }  
}
```

animation-timing-function – altså „easing“

Animation-timing-function – „easing“ – kodes ligesom med transitions, og får virkning på alle animationens keyframes. Som default afvikles animation, ligesom transitions, med en standard easing-in-out-agtig forsinkelse. For at undgå easing imellem alle keyframes i animation kan du sætte animation-timing-function til „linear“, og herefter selv skabe easing i din komposition af keyframes, som omtalt tidligere:

```
#box {  
  -webkit-animation-timing-function: linear;  
  animation-timing-function: linear; }  
}
```

Eller kortere:

```
#box {  
  -webkit-animation: animer_keyframes 4s linear;  
  animation: animer_keyframes 4s linear; }  
}
```

animation-delay

Animation-delay udsætter afspilningen af den samlede animation, og kodes som tal nummer 2 (med en tidsangivelse – „s“ eller „ms“) efter første tal, der jo angiver animationens samlede varighed:

```
#box {  
  -webkit-animation-delay: 10s;  
  animation-tdelay: 10s; }  
}
```

Eller kortere:

```
#box {  
  -webkit-animation: animer_keyframes 4s 10s 4 alternate ;  
  animation: animer_keyframes 4s 10s 4 alternate ; }  
}
```

Nu er der mange tal...

- **første tal**, med tidsangivelse, angiver animationens samlede tid
- **andet tal**, med tidsangivelse, angiver animationens delay
- **tredje tal, uden** tidsangivelse, angiver animationens antal loops