Learn the Web

CSS animations & effects cheat sheet

Transforms · Transitions · Animations · Filters · Target

Transforms

`transform: rotate(deg)`

Rotate an element and any children a certain number of degrees.
Can use negative numbers to go backwards.

```
.dino {
  transform: rotate(-33deg);
}
```

`transform: translate(x, y)`

Similar to `position relative`; will move an element around on the screen without affecting other elements.
The position is based on where the element is currently located.

```
.dino {
  /* Move rightwards 5em and no vertical
  transform: translate(5em, 0);
}
```

Has companion functions to move only in one direction:

```
transform: translateX();
transform: translateY();
```

`transform: scale(factor)`

Grow or shrink an element and all its children.
`1` is what the element currently is; `.6` is smaller; `2.3` is bigger.

```
.dino {
  transform: scale(1.4);
}
```

Has companion functions to scale only in one direction:

```
transform: scaleX();
transform: scaleY();

/* Or combined together */
transform: scale(1.4, 3);
```

`transform: skew(deg, deg)`

Skew an element horizontally and vertically.

```
.dino {
  /* Leaving the second value off will on
  transform: skew(12deg);
}
```

Has companion functions to skew only in one direction:

```
transform: skewX();
transform: skewY();
```

Multiple transforms

Written on a single line, separated by a space.

```
.dino {
  transform: rotate(33deg) scale(1.4);
}
```

Multiple transforms—incorrect example
Multiple lines won't work.
Only the second entry will be activated.

```
.dino {
  /* WRONG */
  transform: rotate(33deg);
  transform: scale(1.4);
}
```

`transform-origin`

Control the anchor point for where the transform occurs.
The default is in the complete centre of the element, aka `center center`
Similar to `background-position`: horizontal then vertical.

```
transform-origin: center center;
/* Top left corner */
transform-origin: left top;
/* Centre of the top edge */
transform-origin: center top;
/* 10px in from the left, 10px down from
transform-origin: 10px 10px;
/* Centre horizontally, 10px up from bott
transform-origin: center calc(100% - 5px)
```

Transitions

Requires user interaction to trigger.

`transition: all 1s linear`

Transition `all` numerical properties that changed.
Lasting `1s`
With `linear` easing (no easing).

```
.dino {
  transition: all 1s linear;
}
```

`transition: background-color 1s linear`

Transition only the `background-color`.

```
.dino {
  transition: background-color 1s linear;
}
```

`transition: all 1s 2s linear`

Delay starting the transition for `2s`

```
.dino {
  transition: all 1s 2s linear;
}
```

Multiple transitions

Written on a single line, separated by a comma.

```
.dino {
  transition: background-color 1s linear,
}
```

Easings

`linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out`
`steps()` — instead of a smooth transition, specific number of frames.

```
.dino {
  transition: background-position 1s step
}
```

Create your own with `cubic-bezier()` — Cubic Bezier Generator

Always on the original state

Do not put `transition` in `:hover` — it won't do what you expect.

```
.dino:hover {
  /* WRONG */
  transition: all 1s linear;
}
```

Animations

Can play automatically or on user interaction.

`@keyframes`

First component of an animation.
Name the keyframes whatever you'd like—following naming conventions.

```
@keyframes wiggle {}
@keyframes dance {}
@keyframes faderoo {}
@keyframes blabidy-boo {}
```

`@keyframes` keywords

Use the start & end keywords.

```
@keyframes wiggle {

  start {
    transform: translateX(-2em);
  }

  end {
    transform: translateX(-4em);
  }

}
```

`@keyframes` percentages

Use percentages to define the different animation keyframes.

```
@keyframes wiggle {

  0% {
    transform: translateX(0em);
  }

  40% {
    transform: translateX(-2em);
  }

  80% {
    transform: translateX(2em);
  }

  100% {
    transform: translateX(0em);
  }

}
```

`animation: wiggle 1s linear`

Use the keyframes set named `wiggle`
Make the animation last `1s`
Have `linear` (no) easing.

```
.dino {
  animation: wiggle 1s linear;
}
```

`animation: wiggle 1s 2s linear`

Delay starting the animation for `2s`

```
.dino {
  animation: wiggle 1s 2s linear;
}
```

`animation: wiggle 1s linear infinite`

`infinite` — Animation iteration count: loop the animation keyframes infinite number of times.
Use a number to choose how many iterations.

```
.dino {
  animation: wiggle 1s linear infinite;
}

.moon {
  /* Play the animation 5 times */
  animation: wiggle 1s linear 5;
}
```

`animation: wiggle 1s linear alternate`

`alternate` — Animation direction: play the keyframes forwards then backwards.
Directions: `normal`, `reverse`, `alternate`, `alternate-reverse`.

```
.dino {
  animation: wiggle 1s linear alternate;
}
```

`animation: wiggle 1s linear forwards`

`forwards` — Animation fill mode: keep the animation on its last frame when complete.
Modes: `forwards`, `backwards`

```
.dino {
  animation: wiggle 1s linear forwards;
}
```

Easings

`linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out`
`steps()` — instead of a smooth transition, specific number of frames

```
.dino {
  animation: wiggle 1s steps(4);
}
```

Create your own with `cubic-bezier()` — Cubic Bezier Generator

Combine multiple options together

```
.dino {
  animation: dance 1s 2s 6 alternate;
}
```

Use the `dance` keyframes, play the animation for `1s`, wait `2s` to start the animation, loop the keyframes `6` times, and `alternate` the keyframe play direction forwards & backwards

`animation on :hover`

Put `animation` in `:hover` to trigger when interacted with

```
.dino:hover {
  animation: dance .3s linear;
}
```

Filters

Be careful with image filters, they're very memory intensive and can slow your website down significantly. See more filters.

`grayscale(%)`

Will desaturate text, elements & images.
`0%` is no change, `100%` is black & white.
Make sure to spell "gray" the American way.

```
filter: grayscale(42%);
```

`blur(px)`

Will blur text, elements & images.
Accepts a pixel number representing the blur radius.

```
filter: blur(7px);
```

`brightness(%)`

Will adjust the brightness of text, elements & images.
`100%` is no change; `0%` is completely black; over `100%` is brighter.

```
filter: brightness(126%);
```

`contrast(%)`

Will adjust the contrast of text, elements & images.
`100%` is no change; `0%` is completely grey; over `100%` is more contrast-y.

```
filter: contrast(78%);
```

`saturate(%)`

Will adjust the colour saturation of text, elements & images.
`100%` is no change; `0%` is completely black & white; over `100%` is more saturated.

```
filter: saturate(258%);
```

`sepia(%)`

Will convert text, elements & images to sepia tones.
`0%` is no change, `100%` is completely sepia.

```
filter: sepia(88%);
```

`drop-shadow(x, y, radius, color)`

Will add a drop-shadow to elements, text & images. It will see inside the image and add a drop-shadow around the non-transparent pixels.
Has the same values as the standard CSS `text-shadow` property.
Needs four properties: *horizontal offset, vertical offset, blur radius, colour.*

```
filter: drop-shadow(2px 2px 10px rgba(0,
```

Multiple filters

Multiple filters can be applied by separating with a space.

```
.dino {
  filter: contrast(120%) grayscale(100%);
}
```

Filters, hover & transition

Since the filters are numerical they can be animated!

```
.dino {
  filter: contrast(120%) grayscale(100%);
  transition: all .2s linear;
}

.dino:hover {
  filter: contrast(100%) grayscale(0);
}
```

Target

`:target`

Style an element when the URL matches the `id` of an element.
URL: `https://dinos-r-us.ca/#stego`

```
<h1 id="stego">Stegosaurus</h1>

#stego {
  background-color: yellow;
}

#stego:target {
  background-color: yellow;
}
```

Target links

```
<ul>
  <li><a href="#stego">Stegosaurus</a></l
  <li><a href="#tri">Triceratops</a></li>
</ul>

<div class="dino" id="stego">…</div>
<div class="dino" id="tri">…</div>

.dino {
  border: 1px solid #e2e2e2;
}

.dino:target {
  border-color: #f33;
}
```

Animate when targeted

```
<a href="#dino">Go Dino, Go!</a>

<img id="dino" src="images/dino.svg" alt=

#dino:target {
  animation: wiggle 1s linear;
}
```


CSS animations & effects

Topics

Advanced SVG

Cheat sheets & checklists

Want a quick review with just a list of all the cheat sheets, checklists & flowcharts on the website? Here it is!

See all the cheat sheets & checklists

© Algonquin College Graphic Design — algonquindesign.ca
All code for this website licensed under the MIT License.
All tutorial code licensed under the CC0 Public Domain Dedication.
All text, imagery, design, tutorials & videos licensed under the CC BY-NC-SA License.
Unless otherwise noted.
Privacy & cookies policy
Algonquin College, 1385 Woodroffe Ave, Ottawa, ON, Canada, K2G 1V8