ScopeTreasure Courses

CSS COURSES · LESSON 2.13 · CODE SCOPE SERIES

CSS Animations Deep — Movement Ki Duniya

Animations beyond hover: @keyframes in depth, delay, direction, fill mode and iteration — movement with full control. Video coming soon.

Introduction

Lesson 2.10 gave your animations an engine; this lesson gives you the **full dashboard**. A real animation is not just "spin" — it has a **delay** before it starts, a **direction** (forwards, backwards, yo-yo), a **fill mode** (what the element looks like before and after), and a **rhythm** across several keyframe stops. Master these six dials and you can choreograph loading screens, entrance effects and living backgrounds — movement that starts, performs and ends exactly where you planned. *(Roman Urdu: Animation ke asli control ye hain — delay, direction, fill mode. Ye seekh liye to harkat aap ke ishare par hogi.)*

Key points

- **The animation property family:** `animation-name` (which @keyframes), `animation-duration` (how long one round takes), `animation-delay` (wait before round one), `animation-iteration-count` (how many rounds — a number or `infinite`), `animation-direction`, `animation-fill-mode`, `animation-timing-function`. The shorthand `animation:` takes them all in one line, in a forgiving order — name, duration, then the rest.
- **`animation-delay` has a trap:** during the delay, the element shows its **normal styles**, not the animation's first frame. A box that should slide in from the left sits visibly *at its final place* during a 1s delay, then jumps and slides. The cure is fill mode (below).
- **`animation-fill-mode` controls the before and after:** `backwards` = during the delay, wear the **first** keyframe; `forwards` = after the last round, **keep** the final keyframe instead of snapping back; `both` = both behaviours. If your entrance animation "flashes then restarts", you wanted `backwards`. If it "snaps back when done", you wanted `forwards`.

  (Roman Urdu: Fill mode batata hai ke animation se pehle aur baad mein element kaisa dikhe.)
- **`animation-direction`:** `normal` (start → end, repeat), `reverse` (end → start), `alternate` (start → end → start → …, the yo-yo — perfect for breathing/pulsing loops so they never jump), `alternate-reverse` (yo-yo starting backwards).
- **Multi-stop keyframes:** frames are not just `from` / `to` — percentages place as many stops as the story needs: `0%` (hidden, low), `60%` (overshoot), `100%` (settle). That overshoot stop is the secret of "bouncy" entrances.
- **Several animations, one element:** comma-separate them — `animation: slide-in 0.6s both, pulse 2s 0.6s infinite;` — an entrance that then breathes forever. And `animation-play-state: paused` (often on `:hover`) freezes any running animation mid-frame.

Code examples

Example 1 — The bouncing ball (percentages + alternate)

```
.ball {
  width: 44px; height: 44px; border-radius: 50%;
  background: #e8490f;
  animation: bounce 0.7s ease-in infinite alternate;
}
@keyframes bounce {
  from { transform: translateY(0); }
  to   { transform: translateY(90px); }
}
```

What it does: One round drops the ball 90px with `ease-in` (gravity's acceleration — slow start, fast fall). `alternate` plays the next round *backwards* — the ball rises with the same curve reversed, which reads as a natural bounce instead of a teleport back to the top.

What you will see: a ball bouncing forever, speeding up as it falls and slowing as it rises. Change `alternate` to `normal` and the lie is exposed: the ball snaps from bottom to top every round. Direction is not a detail; it is the realism.

Example 2 — Delay + fill-mode backwards: the polite entrance

```
.toast {
  background: #0b0f19; color: #faf9f6;
  padding: 14px 20px; border-radius: 10px;
  animation: toast-in 0.5s ease-out 1s both;
}
@keyframes toast-in {
  from { opacity: 0; transform: translateY(16px); }
  to   { opacity: 1; transform: translateY(0); }
}
```

What it does: The toast should appear one second after the page loads (`1s` delay), sliding up and fading in. The last word — `both` — is the professional in the room: `backwards` makes the element wear the `from` frame (invisible, lowered) *during* the delay, and `forwards` keeps it settled afterwards.

What you will see: page loads — no toast for a second — then it rises smoothly into place and stays. Remove `both` and reload: the toast sits fully visible for the first second, vanishes, and slides in — the flash that fill mode exists to prevent.

Example 3 — Two animations and a pause button

```
.dot {
  width: 12px; height: 12px; border-radius: 50%;
  background: #0b6e4f;
  animation: blink 1.2s ease-in-out infinite;
}
.dot:nth-child(2) { animation-delay: 0.2s; }
.dot:nth-child(3) { animation-delay: 0.4s; }
.dot:hover { animation-play-state: paused; }

@keyframes blink {
  0%, 100% { opacity: 0.25; transform: scale(0.85); }
  50%      { opacity: 1;    transform: scale(1.15); }
}
```

What it does: Three identical dots share one keyframe set; staggered `animation-delay` (0 / 0.2 / 0.4s) turns them into the classic "typing…" wave — the delay-as-*offset* trick from Lesson 2.10's practice, done properly. Hovering any dot freezes it mid-pulse via `animation-play-state`.

What you will see: a gentle wave of green dots, left to right, endlessly — and the dot under your mouse held frozen while its brothers keep dancing. One keyframes block choreographed the whole row.

Quick recap

- The six dials: **duration, delay, iteration-count, direction, fill-mode, timing-function** — shorthand `animation:` carries them all.
- **Fill mode**: `backwards` = wear the first frame during delay; `forwards` = keep the last frame after the end; `both` = both.
- **Direction** `alternate` = the yo-yo that makes loops believable; **percentage stops** = multi-act stories; `animation-play-state: paused` = the freeze button.

Common mistakes

1. **Delay without fill-mode backwards** — the element flashes in its final state, then jumps to the first frame and animates. The flash is not a browser bug; it is the missing `both` (or `backwards`).
2. **Expecting `forwards` to freeze mid-animation** — fill mode only dresses the *outside* of the animation (before round one, after the last round). To hold a pose at the end, the animation must *finish* (finite count) — `infinite` never reaches "after".
3. **The same ease for falling and rising** — a `normal`-direction loop replays one curve, so rises look like rewinds. `alternate` + a gravity-flavoured curve is what sells physical motion.

Practice

1. Build a notification banner that enters after a 1.5s delay with `both`, no flash, then gently pulses (second animation) until hovered — where it must pause.
2. Choreograph a three-stop entrance for a card: `0%` invisible + 20px low, `70%` visible + 4px *overshoot* high, `100%` settled. Describe before running why the overshoot stop makes it feel alive.

Practice quiz

1. What does animation-delay do?

- How long one round takes
- The wait before the animation starts
- How many rounds run
- The direction of play

Answer: The wait before the animation starts

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/css-animations-deep · More free tools and notes at https://scopetreasure.com