ScopeTreasure Courses

CSS COURSES · LESSON 2.10 · CODE SCOPE SERIES

Transitions & Animations

Bring pages to life politely: smooth hover transitions, timing functions, @keyframes spinners and badges — and when NOT to move things.

Introduction

A page without motion feels dead; a page where everything bounces feels like a toy shop. Professional motion is a seasoning, not the meal — a button that eases into its hover colour, a spinner that says "loading", a badge that gently pulses for attention. CSS gives you two engines: **transitions** (smooth change between two states, triggered by something — usually hover) and **animations** (self-running keyframe sequences). This lesson drives both, with the timing curves that make motion feel *natural* instead of robotic. *(Roman Urdu: Motion woh namak hai jo page ko zinda karta hai — zyada ho jaye to khana kharab.)*

Key points

- **A transition needs three things:** a property that will change (`background-color`, `transform`…), a **duration** (`0.3s`), and a trigger — most often `:hover`. You declare it on the **base** element: `transition: background-color 0.3s ease;` — then changing that property, by any means, animates smoothly instead of jumping.
- **Timing functions are the* character *of the movement:** `ease` (slow–fast–slow, the default and the most natural), `linear` (robot-constant speed — right for spinners), `ease-in` (starts slow), `ease-out` (lands soft). Two balls with the same duration but different curves arrive feeling completely different.
  (Roman Urdu: Duration waqt hai, timing function us waqt ka mizaj hai.)
- **`transform` is the animator's best friend:** `scale(1.05)` grows, `translateY(-4px)` lifts, `rotate(45deg)` turns — and transforms animate *cheaply* (the browser does not re-lay-out the page), unlike animating `width` or `margin`.
- **Animations run themselves:** you define **@keyframes** — named frames (`from`/`to`, or percentage stops) — then attach them: `animation: spin 1s linear infinite;`. No hover needed; it starts with the page and can loop forever (`infinite`) or a set number of times.
- **Transition vs animation, the table:** transition = two states, needs a trigger, one smooth glide. Animation = many frames, runs on its own, loops, pauses and direction control (deep in Lesson 2.13).
- **Respect the visitor:** keep interface motion short (0.2–0.5s), never animate essential reading, and honour `@media (prefers-reduced-motion: reduce)` — some users get dizzy from motion; their system setting asks you to calm it down, and professionals listen.

Code examples

Example 1 — Jump vs smooth: the 0.3s difference

```
.btn {
  background: #0b6e4f; color: white;
  padding: 12px 22px; border-radius: 10px;
  transition: background-color 0.3s ease, transform 0.3s ease;
}
.btn:hover {
  background: #0e8a63;
  transform: translateY(-3px);
}
```

What it does: The **base** rule carries the transition (this placement is the whole trick). On hover, the colour lightens and the button lifts 3px — over 0.3 seconds, with an ease curve, instead of snapping.

What you will see: hover slowly in and out: the button glides both ways. Now delete the `transition` line and hover again — the same colour change, but it *jumps*. One line is the entire difference between "cheap" and "polished".

Example 2 — A spinner from @keyframes

```
.spinner {
  width: 40px; height: 40px;
  border: 5px solid #e3e6ec;
  border-top-color: #e8490f;
  border-radius: 50%;
  animation: spin 0.9s linear infinite;
}
@keyframes spin {
  to { transform: rotate(360deg); }
}
```

What it does: A circle with a grey border and **one orange segment** (the top border). The keyframes have only a `to` frame — rotate a full turn — because the start (0deg) is wherever it already is. `linear` keeps the speed constant (a spinner that speeds up and slows down looks broken), `infinite` loops it forever.

What you will see: a smooth orange arc chasing itself around the ring — the universal "please wait" of the web, built from one div and two rules. Every loading spinner you have ever seen is this trick wearing different clothes.

Example 3 — A pulse badge and a bouncing arrow

```
.badge {
  background: #e8490f; color: white;
  border-radius: 999px; padding: 4px 10px;
  animation: pulse 1.6s ease-in-out infinite;
}
@keyframes pulse {
  0%, 100% { transform: scale(1); }
  50% { transform: scale(1.12); }
}

.arrow { animation: drop 1.4s ease-in-out infinite; }
@keyframes drop {
  0%, 100% { transform: translateY(0); }
  50% { transform: translateY(8px); }
}
```

What it does: Percentage keyframes give *multi-stop* motion: the badge breathes (normal → 12% bigger → normal), the arrow nudges down and returns — the classic "scroll down" hint. `ease-in-out` softens both ends of each cycle.

What you will see: the badge gently swelling beside a heading — noticeable, not annoying — and the arrow bobbing under the hero. Set either duration to 0.4s and feel how *rude* it becomes: speed is a design decision.

Quick recap

- **Transition**: two states + trigger + duration — declare it on the **base** element. **Timing functions** give motion its character; `transform` animates cheapest.
- **Animation**: @keyframes frames + `animation: name duration timing iteration` — runs and loops by itself.
- Short, purposeful, reducible: 0.2–0.5s for interface motion, and respect `prefers-reduced-motion`.

Common mistakes

1. **Putting the transition inside `:hover`** — the hover state animates *in*, then snaps *back* instantly when the mouse leaves, because the base state has no transition. The transition always lives on the base rule.
2. **Animating width, height or margin for movement** — the browser re-lays-out the page every frame (jank on weak phones). Move things with `transform`, fade with `opacity` — the two cheap properties.
3. **Motion everywhere** — five pulsing, bouncing, spinning elements compete until the page reads as noise. One or two purposeful motions per screen is the professional budget.

Practice

1. Build a hover gallery of four cards, each with a *different* transition: colour fade, lift (translateY + shadow), grow (scale), and underline-slide. All share `transition: all 0.3s ease` — then set one card's duration to 1.2s and feel the difference.
2. Make the Example 2 spinner, then a "Loading…" row of three dots that pulse in sequence (you will need `animation-delay` — Lesson 2.13's first tool — try 0s / 0.2s / 0.4s).

Practice quiz

1. Where must a transition be declared?

- Inside the :hover rule
- On the base element, so it animates in and out
- Only inside @keyframes
- In the HTML, never in CSS

Answer: On the base element, so it animates in and out

2. How is an animation different from a transition?

- Animations need a hover trigger, transitions do not
- An animation runs by itself from @keyframes frames, with no trigger needed
- Animations cannot loop
- There is no difference at all

Answer: An animation runs by itself from @keyframes frames, with no trigger needed

3. Which property animates cheaply, without re-laying-out the page?

- width
- margin
- transform
- padding

Answer: transform

Watch the video lesson: https://youtu.be/tmooa0i7hAI

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