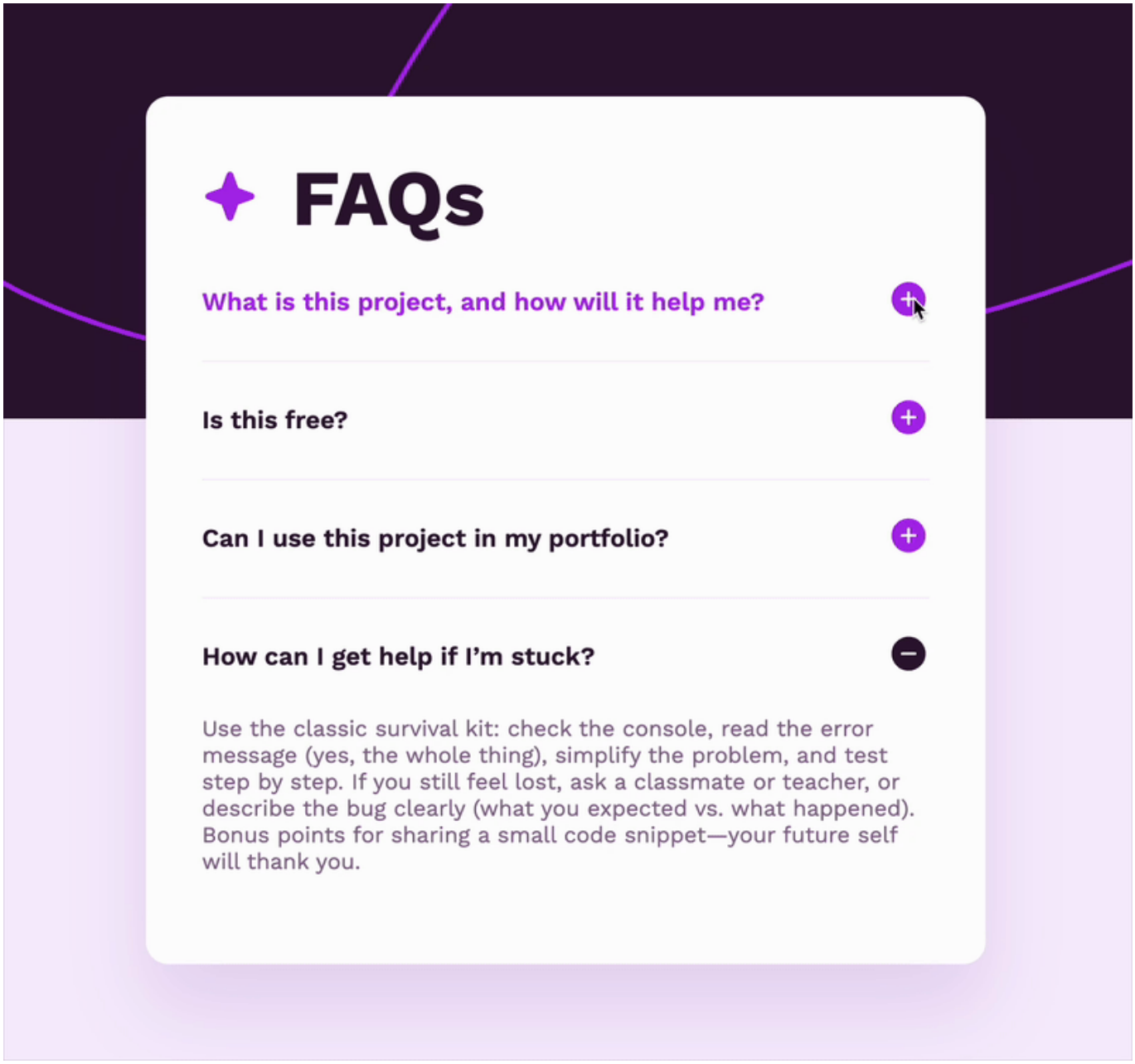
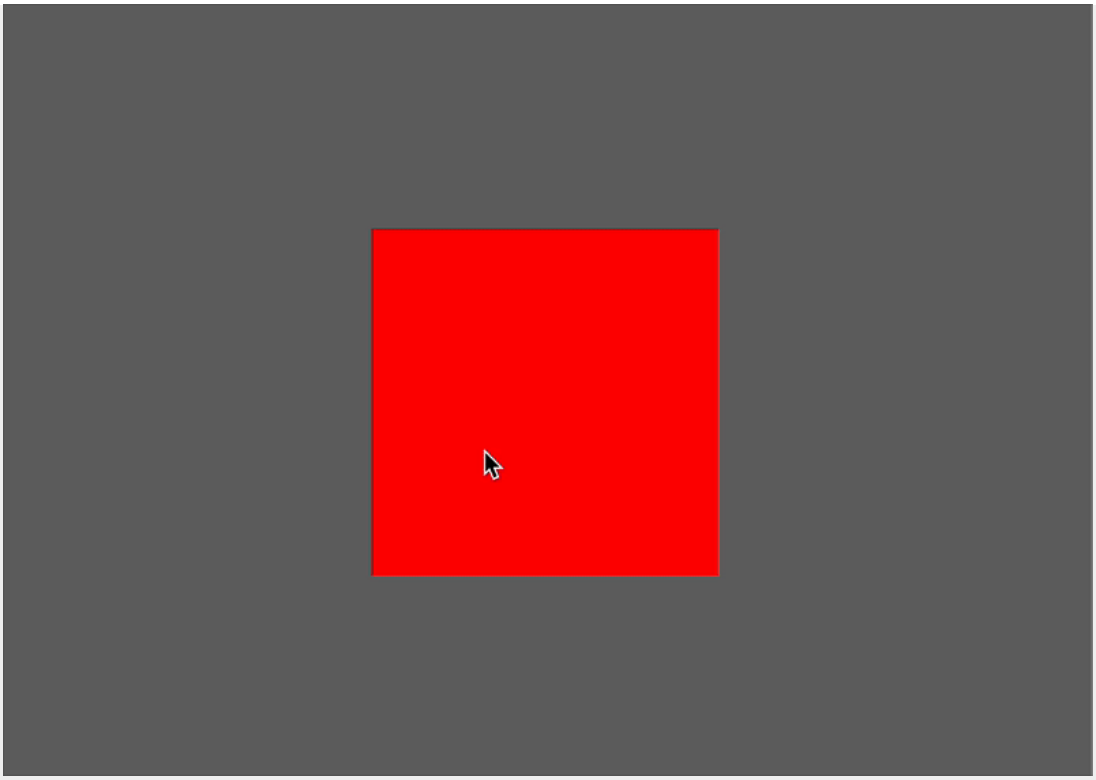
LU06a – CSS Transitions



CSS Transitions – Grundprinzip

Eine CSS Transition animiert den Übergang zwischen zwei Zuständen eines Elements. Sie definieren, **welche Eigenschaft** animiert wird, **wie lange** die Animation dauert und nach welcher **Timing-Kurve**.



Hover-Transition der Hintergrundfarbe

```
.element {
  background-color: blue;
  transition: background-color 400ms ease-in-out;
}

.element:hover {
  background-color: red;
}
```

Beim Hovern wechselt die Hintergrundfarbe nun nicht abrupt, sondern gleitend über 400 ms von Blau zu Rot.

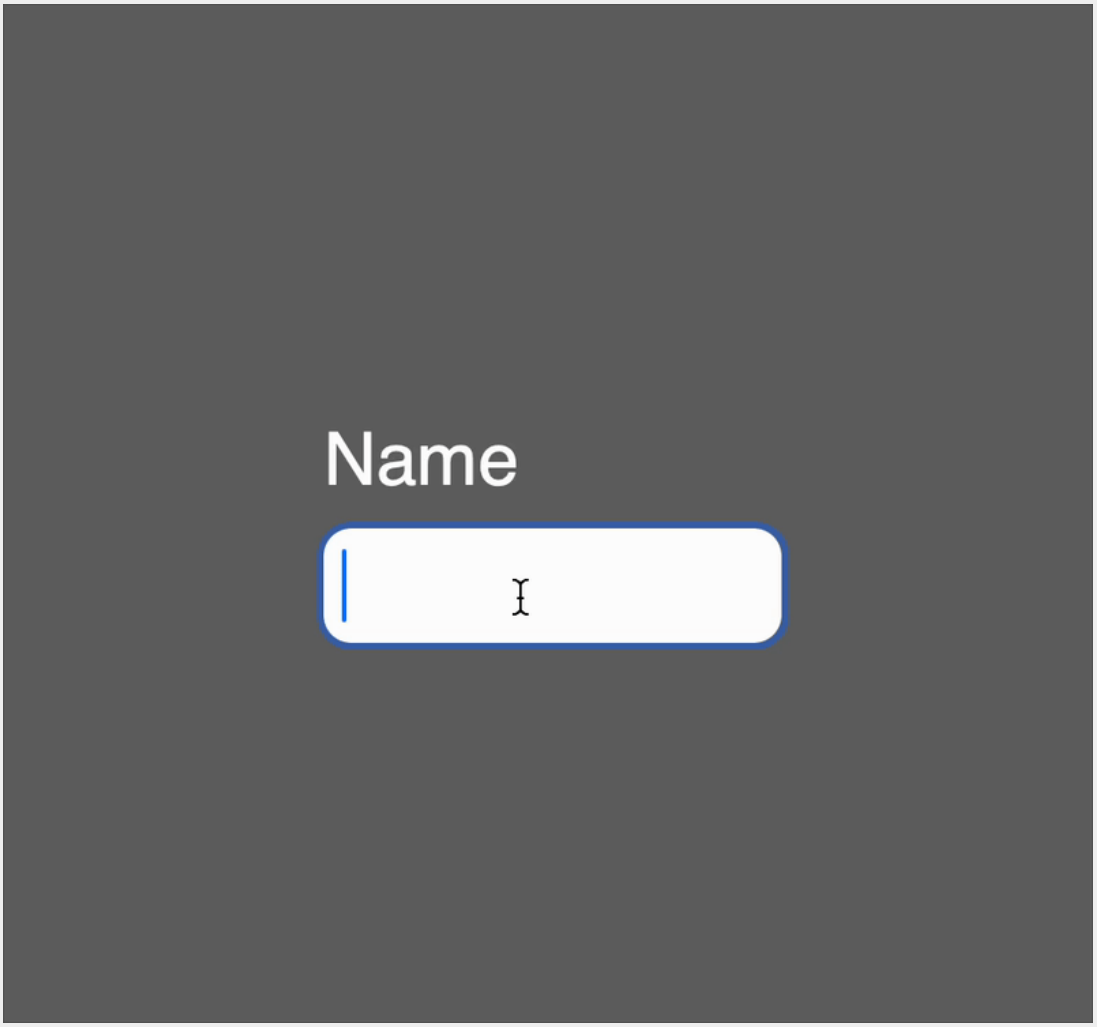
Mehrere Eigenschaften gleichzeitig animieren:

```
.panel {
  opacity: 0;
  height: 0;
  transition:
    opacity  600ms ease-in-out,
    height   600ms ease-in-out;
}
```

Trigger: Was löst eine Transition aus?

Damit eine Transition aktiviert wird, braucht es zwei Dinge: eine **Eigenschaft, die sich ändert**, und ein **Ereignis, das diese Änderung auslöst**. Die häufigsten Trigger in CSS:

Trigger	Beschreibung
`:hover`	Cursor befindet sich über dem Element
`:focus`	Element ist fokussiert (z.B. per Tab-Taste)
`:focus-within`	Element oder eines seiner Kind-Elemente ist fokussiert
`:active`	Element wird gerade aktiviert (Maustaste gedrückt)
`:target`	URL-Fragment stimmt mit der `id` des Elements überein
Klasse via JavaScript	Eine CSS-Klasse wird per `classList.add/remove/toggle` geändert



Durch Klicken ins Formular-Feld erhält das Element den Fokus → Ein Trigger für eine Transition der Transparenz.

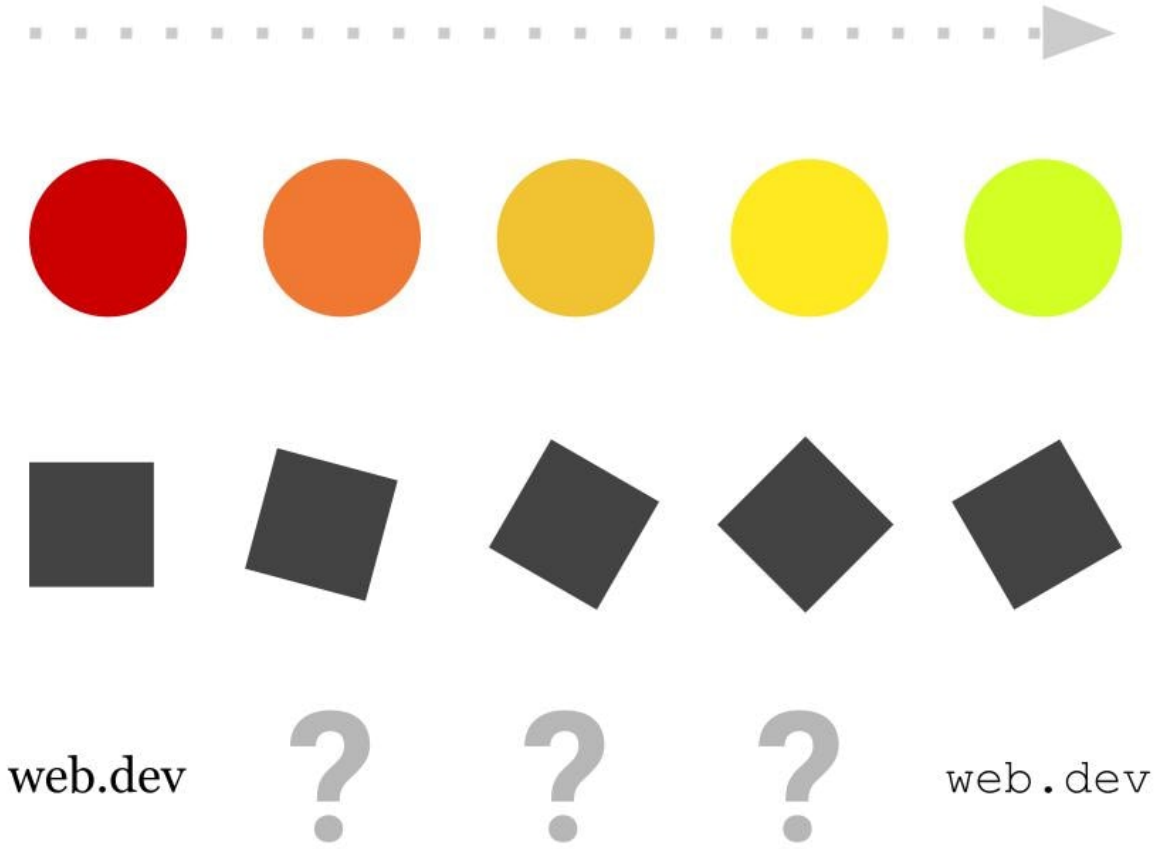
Der letzte Punkt ist für unser Accordion entscheidend: Wir setzen per JavaScript die Klasse `open` – und CSS übernimmt die Animation vollständig. JavaScript steuert den **Zustand**, CSS steuert die **Animation**.

Was kann (und was kann nicht) animiert werden?

Nicht jede CSS-Eigenschaft lässt sich animieren. Die Grundregel: Eine Eigenschaft ist animierbar, wenn zwischen Start- und Endwert ein **Zwischenwert berechnet werden kann**.

Beispiel: `font-size: 12px` → `font-size: 24px` – jeder Pixelwert dazwischen ist berechenbar.

Gegenbeispiel: `font-family: serif` → `font-family: monospace` – was wäre eine Schriftart-Mitte?

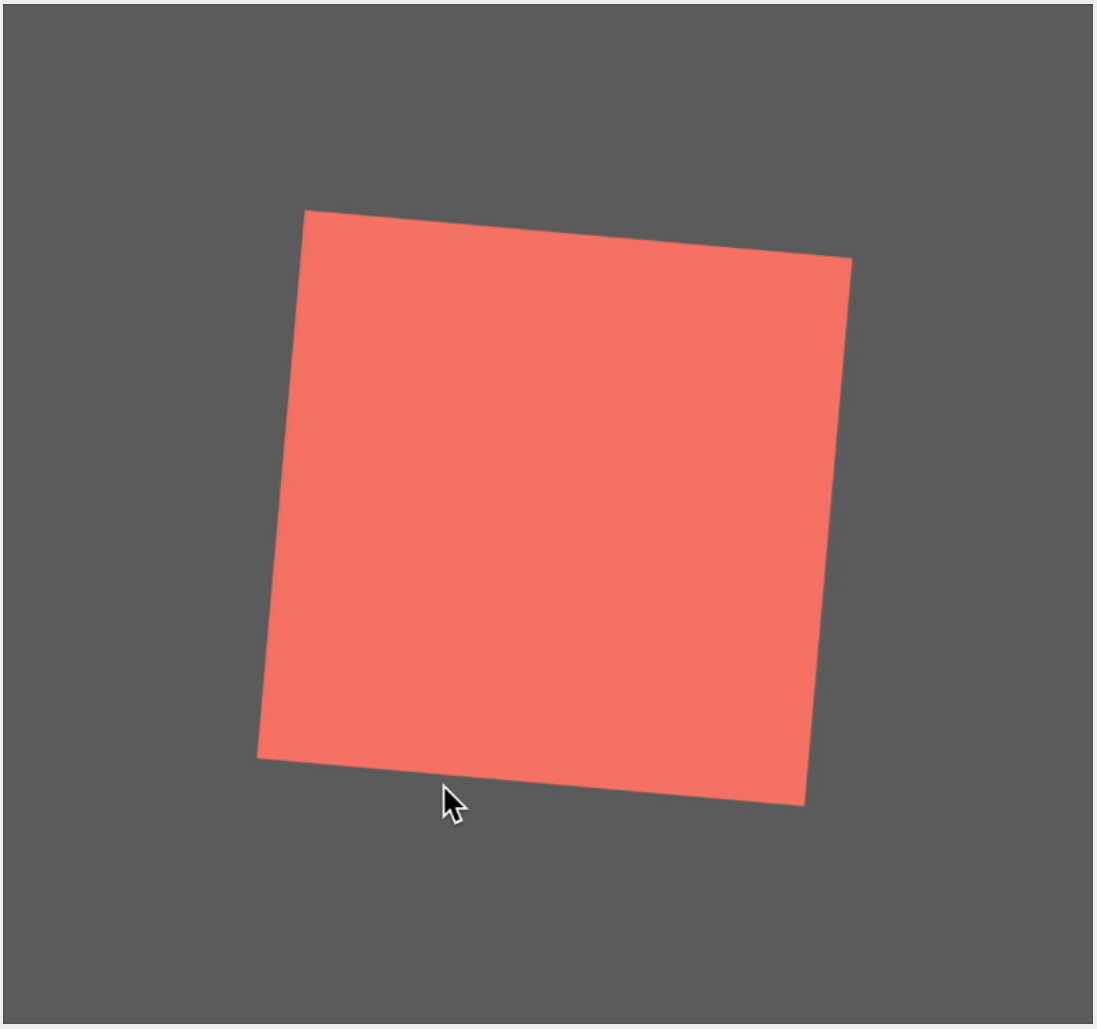


Gut animierbare Eigenschaften

transform

`transform` ist die am häufigsten eingesetzte Eigenschaft für Transitions. Sie wird direkt von der **GPU**

berechnet – das ergibt flüssigere Animationen und belastet den Browser weniger als Layout-Eigenschaften.



```css
.box {
  transition: transform 400ms ease-out;
}

.box:hover {
  transform: scale(1.2) rotate(5deg) translateY(-8px);
}
```

Einzelne Transform-Eigenschaften können auch separat animiert werden:

```css
.box {
  transition:
    scale     300ms ease-out,
    rotate    300ms ease-in-out,
    translate 300ms ease-out;
}
```

```css
.box:hover {
  scale: 1.2;
  rotate: 5deg;
  translate: 0 -8px;
}
```

Farbe

`color`, `background-color` und `border-color` sind häufige Transition-Kandidaten – sie geben bei Interaktion visuelles Feedback, z.B. wenn ein Button beim Hovern die Farbe wechselt.

```css
.btn {
  background-color: steelblue;
  transition: background-color 250ms ease-in-out;
}

.btn:hover {
  background-color: royalblue;
}
```

opacity

Sanftes Ein- und Ausblenden – das Element **bleibt aber im Layout** (es nimmt weiterhin Platz ein).

Weitere animierbare Eigenschaften

Eigenschaft	Einsatz
`width`, `height`	Zwischen zwei fixen Werten (z.B. `0` → `200px`)
`margin`, `padding`	Layout-Animationen
`border-radius`	Formen weich überblenden
`box-shadow`	Elevation-Effekte (z.B. bei Fokus)
`filter`	`blur()`, `brightness()`, `saturate()` etc.
`font-size`, `letter-spacing`	Typografische Animationen

Nicht animierbar

Eigenschaft	Warum nicht?
`display`	Kein Mittelwert zwischen `none` und `block` möglich
`font-family`	Kein Mittelwert zwischen Schriftarten möglich
`position`	Kategoriewechsel, kein kontinuierlicher Übergang
`visibility`	Binärer Wert (sichtbar/unsichtbar)

Eine vollständige Liste animierbarer Eigenschaften: MDN – Animatable CSS properties

Antworten Ein- und Ausblenden mit 'display'

Im ersten Schritt haben wir die Panels mit `display: none` / `display: block` ein- und ausgeblendet. Das funktioniert – aber es gibt **keinen Übergang**, das Element erscheint und verschwindet abrupt.

Warum lässt sich `display` nicht animieren? CSS Transitions interpolieren zwischen zwei Zahlenwerten. `display: none` bedeutet: das Element **existiert im Layout nicht**. Zwischen „existiert nicht“ und „existiert“ kann kein Mittelwert berechnet werden – eine Transition ist daher logisch nicht möglich.

```
/* ❌ Kein Übergang möglich */
.panel {
  display: none;
  transition: display 400ms; /* wirkungslos */
}
.panel.open {
  display: block; /* schaltet sofort um */
}
```

Das Problem: height von 0 zu auto

Der einfachste und natürlichste Ansatz wäre: `height: 0` → `height: auto` (→ auto: der Browser berechnet selbst die Höhe des Elements anhand der Inhalte). Leider funktioniert das **nicht** mit einer Transition:

```
/* ❌ Funktioniert NICHT – Browser kann nicht interpolieren
*/
.panel {
  height: 0;
```

```
  overflow: hidden;
  transition: height 600ms ease-in-out;
}
.panel.open {
  height: auto; /* Browser weiss nicht, was das in px
bedeutet */
}
```

Der Browser kann zwischen einem fixen Pixelwert (`0`) und dem Schlüsselwort `auto` nicht interpolieren – `auto` ist kein Zahlenwert, sondern eine Anweisung: „berechne die Höhe selbst“.

Die moderne Lösung: interpolate-size: allow-keywords

Seit 2024 unterstützen moderne Browser eine neue CSS-Eigenschaft, die genau dieses Problem löst: `interpolate-size: allow-keywords`.

Sie erlaubt dem Browser, Übergänge zwischen einem Pixel-Wert und einem CSS-Schlüsselwort wie `auto`, `min-content` oder `max-content` zu berechnen.

```
:root {
  interpolate-size: allow-keywords; /* einmal global
definieren */
}
```

Damit funktioniert `height: 0` → `height: auto` mit einer Transition wie erwartet.

CSS-Transitions: Ein- und Ausblenden der Antworten

```
:root {
  interpolate-size: allow-keywords;
}

/* Standardmässig versteckt */
.panel {
  height: 0;
  opacity: 0;
  overflow: hidden;
  transition:
    height  800ms ease-in-out,
    opacity 800ms ease-in-out;
}
```

```css
/* Sichtbar wenn Klasse 'open' gesetzt */
.panel.open {
  height: auto;
  opacity: 1;
}
```

Das Öffnen und Schliessen wird weiterhin per JavaScript durch `classList.add('open')` / `classList.remove('open')` gesteuert – die Animation läuft vollständig in CSS.

Browser-Support: `interpolate-size: allow-keywords` wird aktuell **nur von Chrome/Edge** unterstützt – Safari und Firefox kennen sie noch nicht. In diesen Browsern springt das Panel beim Öffnen abrupt auf `height: auto`, ohne Animation.

Aktuellen Support prüfen: caniuse.com – interpolate-size

Timing-Funktionen

Der dritte Parameter von `transition` steuert die Beschleunigungskurve der Animation:

Wert	Beschreibung
`ease`	Langsam starten, beschleunigen, langsam enden (Standard)
`ease-in`	Langsam starten, schnell enden
`ease-out`	Schnell starten, langsam enden – oft natürlicher
`ease-in-out`	Langsam starten und enden, in der Mitte schnell
`linear`	Konstante Geschwindigkeit
`cubic-bezier(x,x,x,x)`	Vollständig selbst definierte Kurve

Im Accordion-Projekt verwenden wir `ease-in-out` – das Panel öffnet und schliesst sich jeweils mit einer leichten Verzögerung am Anfang und Ende, was natürlicher wirkt.

Interaktives Beispiel mit Beispiel-Code: Externe Seite erkunden.