LU12b - FAQ-Accordion mit Animationen erweitern

Sie haben die Theorie zu CSS Keyframe Animationen gelesen. Jetzt wenden Sie das Gelernte direkt an: Sie erweitern Ihr bestehendes FAQ-Accordion mit drei Animationen, die die User-Experience verbessern.

Merkmal	**Beschreibung**
Zeit	20–25 Minuten
Sozialform	Einzelarbeit – Austausch und Hilfe mit Sitznachbarn erlaubt
Voraussetzung	Accordion lädt Daten von MockAPI (LU11 / LU12 Phase 1 abgeschlossen)

Übersicht: Was wir bauen

Am Ende dieses Auftrags hat Ihr Accordion drei Animationen:

Animation	**Datei**	**Wann**
Spinner	`Accordion.vue`	Während die Daten von der API laden
Fade-in mit Treppeneffekt	`Accordion.vue`	Wenn die FAQ-Items nach dem Laden erscheinen
Einblenden	`AccordionItem.vue`	Wenn eine Antwort aufgeklappt wird

Teil A – Ladeanimation: Spinner

A1 – Keyframes und CSS definieren

Öffnen Sie `Accordion.vue`. Fügen Sie im `<style>`-Bereich folgende Styles ein:

```
/* Spinner-Animation */
@keyframes spin {
  to {
    transform: rotate(360deg);
  }
}

.loading-wrapper {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 0.75rem;
  padding: 2rem 0;
```

```
  color: #888;
  font-size: 0.9rem;
}

.loading-spinner {
  width: 36px;
  height: 36px;
  border: 4px solid #e0e0e0;
  border-top-color: #444;
  border-radius: 50%;
  animation: spin 0.75s linear infinite;
}
```

> **Warum kein `from`?** Der Spinner startet bei `rotate(0deg)` — das ist der CSS-Default. Den Startwert müssen wir nicht explizit schreiben (Partial Keyframes Trick aus der Theorie).
>
> **Warum `linear`?** Bei einem Spinner würde `ease-in-out` das Drehen ungleichmässig wirken lassen — als würde er stocken und beschleunigen. `linear` hält die Drehung konstant flüssig.

A2 – Template anpassen

Ersetzen Sie die bisherige Ladeanzeige „`Daten werden geladen…`„ durch den animierten Spinner:

```
<!-- Vorher: -->
<div v-if="isLoading">Daten werden geladen...</div>

<!-- Nachher: -->
<div v-if="isLoading" class="loading-wrapper">
  <div class="loading-spinner"></div>
  <span>FAQs werden geladen…</span>
</div>
```

A3 – Testen

Öffnen Sie DevTools (F12) → Reiter **Network** → Geschwindigkeit auf **Slow 3G** stellen. Laden Sie die Seite neu. Der Spinner sollte sichtbar sein, bevor die FAQ-Items erscheinen. Stellen Sie die Geschwindigkeit anschliessend wieder zurück.

Teil B – Fade-in mit Treppeneffekt: FAQ-Items erscheinen nacheinander

Wenn die Daten geladen sind und die Items erscheinen, wirkt es eleganter, wenn sie nicht alle gleichzeitig aufpoppen — sondern leicht versetzt nacheinander eingeblendet werden.

B1 – CSS für den Einblende-Effekt

Fügen Sie in `Accordion.vue` → `<style>` hinzu:

```
/* Fade-in Keyframe */
@keyframes fadeInUp {
  from {
    opacity: 0;
    transform: translateY(12px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

/* Wird auf jedes AccordionItem angewendet */
.faq-item {
  animation: fadeInUp 0.4s ease both;
  animation-delay: calc(var(--i) * 70ms);
}
```

> **`both` statt `forwards`:** Weil wir einen `animation-delay` gesetzt haben, würden die Items ohne `both` kurz mit `opacity: 1` aufblitzen, bevor ihre Animation startet. `both` stellt sicher, dass sie bereits während der Wartezeit im Startzustand (`opacity: 0`) bleiben.

B2 – Index als CSS-Variable übergeben

Damit der gestaffelte Delay funktioniert, muss jedes Item wissen, an welcher Position es in der Liste steht. Übergeben Sie den Index aus `v-for` als CSS Custom Property:

```
<!-- Vorher: -->
<AccordionItem
  v-for="item in faqItems"
  :key="item.id"
  :faq="item"
/>

<!-- Nachher: -->
<AccordionItem
  v-for="(item, index) in faqItems"
  :key="item.id"
  :faq="item"
  class="faq-item"
  :style="{ '--i': index }"
/>
```

B3 – Kontrolle

Laden Sie die Seite neu (Slow 3G hilft auch hier). Die FAQ-Items sollten nun nacheinander von unten eingeblendet werden — jedes leicht nach dem vorherigen.

Fragen zum Verstehen:

- Was passiert, wenn Sie `70ms` auf `200ms` ändern? Und auf `0ms`?
- Was bedeutet `calc(var(–i) * 70ms)` konkret für das dritte Item (Index 2)?

Teil C – Einblenden der Antwort beim Öffnen

Das Öffnen eines Accordion-Items soll ebenfalls animiert sein: Die Antwort soll sanft eingeblendet werden, statt abrupt zu erscheinen.

C1 – Keyframe in AccordionItem.vue

Öffnen Sie `AccordionItem.vue`. Fügen Sie im `<style>`-Bereich hinzu:

```
@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(-6px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

.accordion-body {
  animation: fadeIn 0.25s ease forwards;
}
```

> **Kleiner `translateY`-Wert:** Der Text bewegt sich beim Öffnen leicht von oben nach unten — das gibt der Bewegung eine Richtung, die zum Aufklappen passt.

C2 – Kontrolle

Klicken Sie auf ein FAQ-Item. Der Antwort-Text sollte nun weich eingeblendet werden.

Frage: Was passiert ohne `forwards`? Probieren Sie es aus, indem Sie `forwards` entfernen und die Animation beobachten.

Bonus – Bounce Dots statt Spinner

Als Alternative zum Spinner können Sie drei springende Punkte als Ladeanzeige verwenden. Ersetzen Sie in `Accordion.vue` die Spinner-Styles und das Template:

```
@keyframes bounce {
  0%, 80%, 100% { transform: translateY(0); }
  40%           { transform: translateY(-10px); }
}

.loading-dots {
  display: flex;
  gap: 8px;
  justify-content: center;
  padding: 2rem 0;
}

.loading-dots span {
  width: 10px;
  height: 10px;
  border-radius: 50%;
  background: #888;
  animation: bounce 1.2s ease-in-out infinite;
}

.loading-dots span:nth-child(2) { animation-delay: 0.2s; }
.loading-dots span:nth-child(3) { animation-delay: 0.4s; }
```

```
<div v-if="isLoading" class="loading-dots">
  <span></span>
  <span></span>
  <span></span>
</div>
```

Frage: Warum sind die Keyframes `0%` und `100%` identisch? Was würde passieren, wenn `100%` auf `translateY(-10px)` gesetzt würde?

Unter folgendem Link können Sie meine finale Umsetzung des Accordions (Accordion.vue und AccordionItem.vue) herunterladen und mit Ihrem eigenen Code vergleichen.

Finale Accordion-Komponenten

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/m291/learningunits/lu12/aufgaben/a_keyframes

Last update: **2026/05/18 10:20**