LU04b – Events & State

Events & EventListener

Ein **Event** ist ein Signal, das der Browser aussendet, wenn etwas passiert – ein Klick, eine Tastatureingabe, ein Scroll. Mit einem **EventListener** registrieren Sie eine Funktion, die auf dieses Signal reagiert.

Verbindung zum Alarado-Projekt: Beim Dark/Light Toggle haben Sie bereits `addEventListener` verwendet: `checkbox.addEventListener('change', toggleTheme)`. Heute nutzen wir dasselbe Muster für das Accordion.

addEventListener – Grundprinzip

```
const btn = document.querySelector('#mein-button');

// Syntax: element.addEventListener(eventTyp, funktion)
btn.addEventListener('click', function() {
  console.log('Button wurde geklickt!');
});

// Kürzer mit Arrow Function:
btn.addEventListener('click', () => {
  console.log('Geklickt!');
});
```

Wichtige Event-Typen

Event	Wann wird es ausgelöst?	Typischer Einsatz
`click`	Maustaste gedrückt + losgelassen	Buttons, Toggle, Accordion
`keydown`	Taste gedrückt (jede Taste)	Keyboard-Navigation
`keyup`	Taste losgelassen	Formular-Validierung
`change`	Wert eines Inputs geändert + Fokus verloren	Checkbox, Select
`focus`	Element erhält Fokus	Formular, Accessibility
`blur`	Element verliert Fokus	Validierung beim Verlassen
`mouseover`	Maus betritt das Element	Hover-Effekte

Das Event-Objekt

Jeder Handler erhält automatisch ein **Event-Objekt** als Parameter – darin stecken nützliche Informationen:

```
btn.addEventListener('click', (event) => {
  // event.target → das Element, das geklickt wurde
  console.log(event.target);

  // event.key → bei Tastaturevents: 'Enter', 'Escape', ' ' etc.
  // event.preventDefault() → Standardverhalten verhindern (z.B. Formular-
Submit)
});

// Keyboard: Enter und Space wie Click behandeln
btn.addEventListener('keydown', (event) => {
  if (event.key === 'Enter' || event.key === ' ') {
    // gleiche Aktion wie beim Click
  }
});
```

> **Tipp:** Wenn Sie <button> verwenden (nicht), reagiert es von sich aus auf Enter und Space. Sie brauchen dann keinen eigenen Keyboard-EventListener.

State – Zustand im UI

State (Zustand) ist der aktuelle Stand Ihrer Benutzeroberfläche. Beim Accordion hat jedes Panel zwei mögliche Zustände: **offen** oder **geschlossen**.

Das Muster kennen Sie bereits vom Toggle:

Zustand	Klasse gesetzt?	aria-expanded
geschlossen	.panel (kein .open)	„false“
offen	.panel.open	„true“

State aus dem DOM lesen

Sie können den Zustand direkt aus dem DOM ablesen, ohne eine eigene Variable zu führen:

```
const panel = document.querySelector('.panel');
const btn = document.querySelector('.accordion-btn');

btn.addEventListener('click', () => {
  // Zustand aus dem DOM lesen
  const istOffen = panel.classList.contains('open');

  if (istOffen) {
    panel.classList.remove('open');
    btn.setAttribute('aria-expanded', 'false');
  } else {
    panel.classList.add('open');
```

```
    btn.setAttribute('aria-expanded', 'true');
  }

  // Oder kürzer in einem Schritt:
  // panel.classList.toggle('open');
});
```

Alle anderen Panels schliessen

Bei einem klassischen Accordion soll immer nur ein Panel offen sein. Beim Öffnen eines neuen Panels werden alle anderen geschlossen:

```
const buttons = document.querySelectorAll('.accordion-btn');

buttons.forEach(btn => {
  btn.addEventListener('click', () => {
    const panel = btn.nextElementSibling; // das nächste Geschwister-Element
    const istOffen = panel.classList.contains('open');

    // Alle Panels schliessen
    buttons.forEach(andererBtn => {
      andererBtn.nextElementSibling.classList.remove('open');
      andererBtn.setAttribute('aria-expanded', 'false');
    });

    // Dieses Panel öffnen (nur wenn es vorher zu war)
    if (!istOffen) {
      panel.classList.add('open');
      btn.setAttribute('aria-expanded', 'true');
    }
  });
});
```